Calibration of Stereo Images using OTV Correspondences

In partial fulfilment of the M.Phil. Degree

by

Sai-kee Wong

Department of Systems Engineering
The Chinese University of Hong Kong

June 1994

Acknowledgments

This research was supported by Hong Kong Universities and Polytechnics Grants Committee (UPGC) under the CUHK 1992-1994 Block Grant.

Contents

List Of Figures

List Of Tables

Abstract

Stereo images have to be calibrated before stereo vision can recover the three-dimensional information of the imaged scene. Position constraints via point correspondences are traditionally used to solve the calibration problem. We describe a method that uses angle constraints instead. In particular, we use correspondences of projections of orthogonal trihedral vertices, some common features in indoor and outdoor scenes, for calibration. The method has a closed-form solution, as opposed to many other calibration methods which are iterative. It also requires only two vertex correspondences at minimum to recover all the transformation parameters which are recoverable from a stereo image pair. Extensive experimental results are presented, and they show that our method is more robust than those using position constraints alone, recovering more accurate calibration parameters with only a small number of feature correspondences. Unlike those of other methods, the solution is also generally insensitive to the direction and magnitude of the displacement between the stereo views. Performance of our method on real images is also presented.

Chapter 1

Introduction

Sensors that can recover three-dimensional (3-D) information of a scene have tremendous applications in various areas such as industrial inspection, manufacturing automation, and autonomous navigation. Visual sensors are particularly attractive as they can recover 3-D information at relatively long distances. Stereo vision [4, 9, 6] is one of the simplest ways to recover 3-D information directly from two images. It is also a passive means, *i.e.,* it does not require sending out signals that may disturb the objects and the environment.

The idea of stereo vision is simple. Pictures of a scene are first taken at two different viewpoints. Knowing which image points in the two pictures are projected by the same feature in space, which is achieved via a stereo matching process, the position of the feature in 3-D can be located at the intersection of the two lines of sight through the corresponding image points. Such a depth estimation process is known as the triangulation process.

However, stereo pictures have to be calibrated before the stereo matching and the triangulation processes can recover 3-D information. The relative orientation between the two pictures have to be known before the lines of sight can be extrapolated and intersected. This problem is known as the *calibration* problem.

1.1 Motivation

Traditionally, point correspondences over the two images are used to compute the transformation parameters. This can be regarded as using position constraints, as the idea is to move the two views relatively to each other until the lines of sight of the point correspondences intersect in the 3-D space. Yet using the minimum number of point correspondences for a unique interpretation is

found to be inadequate; the solution suffers greatly from quantization noise and other disturbances in the image coordinates of the point correspondences. People have used more than the minimum input data set, but a large limitation of accuracy is still observed. A further step logically is the use of more abstract features over the two views, an example of which is the line segment, so that richer information of the features reduces the effect of noise. Unfortunately, as projection planes of two image lines generally intersect regardless of the camera orientations, use of line correspondences requires more than two views to estimate the transformation parameters. The question becomes: what else can we exploit from a pair of stereo views that would allow us to go beyond the current limitation of accuracy in estimating calibration parameters? That is what we address in this paper.

1.2 Objective of the Study

We propose to use angular relationships among extended features, in addition to position constraints, to estimate the transformation parameters. We believe calibration via angle constraints is even more robust than that via position constraints. For one thing, while position information of point features are subject to quantization effect directly, angular information do not suffer as much from quantization error, as angular properties of extended features are computed over linked lists of edge elements. More importantly, under angle constraints not only the lines of projection of the image features are required to intersect in 3-D, the projection planes going through the extended image features also have to intersect to form lines which are at a fixed angular relationships with one another. The process is robust as it only takes a small disturbance in the relative position of the two views to distort the angular relationship largely.

Since orthogonal trihedral vertices (OTVs) are commonly occuring features in indoor and outdoor scenes, we use angular properties of OTV correspondences projected from objects either originally in the scene or artificially placed there for calibration. We believe by *enforcing* the orthogonality property of the visible OTVs in the final solution, the effect of noise to the calibration parameters can be kept to a minimum.

We use the pinhole camera model for the camera projection processes, and we assume the intrinsic parameters such as the focal length and the pixel (short for picture element) width and height of each camera are known. Our

concern here is to estimate the extrinsic parameters in the stereo system, or more precisely, the rotational R and the translational $\vec{t}$ relationship between the stereo cameras.

1.3 Our Approach

Most methods proposed so far for direct estimation of the relative orientation of two images use position constraints of image features, primarily those of point correspondences. Typically a lot of input correspondences are needed so that a least-squares method would reduce the solution error due to quantization and other noise in the input data. Still, the relative orientation parameters estimated are often found to be not accurate enough. The reason is, all the method does is to move the two image coordinate frames one relative to the other, until all the lines of projection through the corresponding points intersect in 3-D (Figure 1.1). Such a final configuration is an extremum of the solution space. The problem is, such an extremum is not a sharp one, as at the extremal configuration we can easily slide one view relative to the other along the lines of projection, while keeping the lines more or less intersecting in the 3-D space.

In view of this, we propose to use angle constraints as opposed to position constraints alone for calibration. In particular, we use the angular property of orthogonal trihedral vertices (OTVs) which are trihedral vertices with edges orthogonal to one another. OTVs are common features in indoor and outdoor scenes, and are especially abundant in aerial scenes. The image of a typical building like the one shown in Figure 1.2 already contains four OTVs whose branches are all visible. We believe solution via angle constraints is more stable, as not only the lines of projection of the image features are required to intersect in 3-D, the projection planes going through the extended image features also have to intersect to form lines which are at a fixed angular relationship with one another. At the extremal configuration, small disturbance in the relative position of the two views will already distort the angular relationship largely. We believe by *enforcing* the orthogonality property of the visible OTVs in the final solution, the effect of noise to the calibration parameters can be minimized.

OTVs have been used very often as exploitable features in various domains like stereo and motion analyses, and especially in aerial image understanding [20, 3, 12]. Surprisingly, very few [18, 34] ever looked at whether the orientation

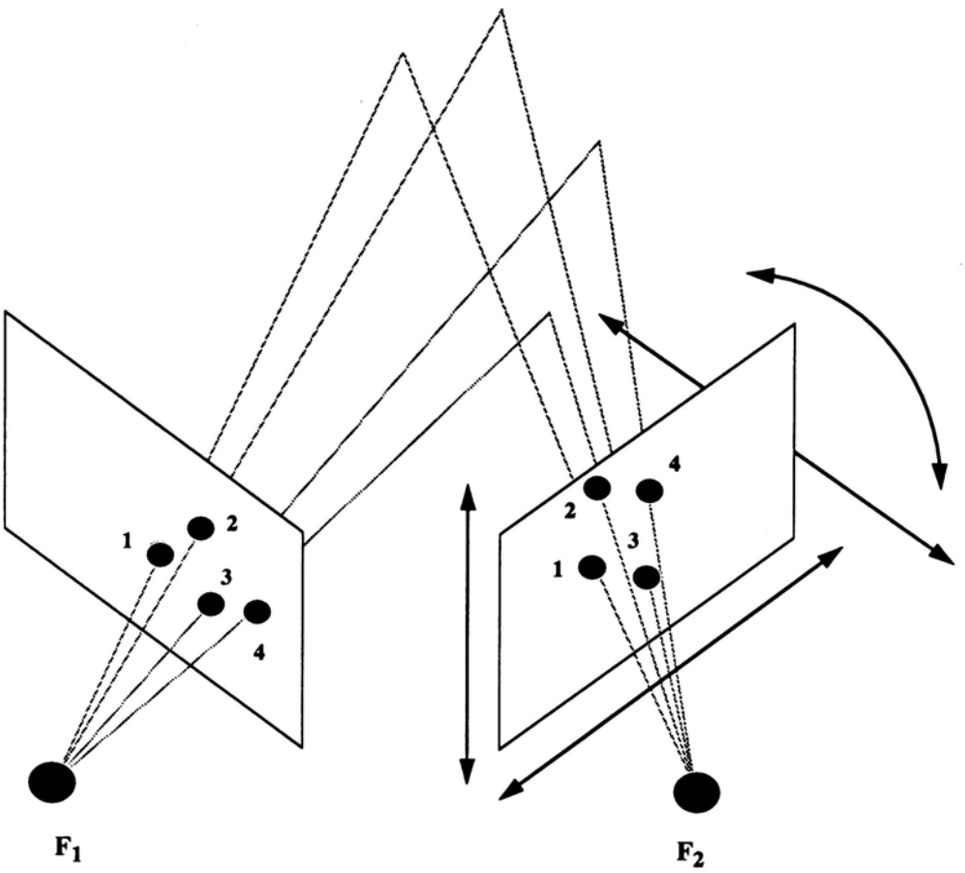


Figure 1.1: Stereo calibration using point correspondences.

of an OTV is recoverable from its image or not, not mentioning how the analysis can be exploited in aerial image understanding. In fact Kanatani [18] proposed a method which makes use of constraints on length and angle known *a priori* to recover the orientation of a trihedral vertex. Wu *et al.* [34] also extended Kanatani's work and worked out a closed-form solution from angle to angle correspondences.

1.4 Original Contributions

The major contributions presented in this thesis are:

- Developed a simple method to recover the orientation of three branches of an OTV up to two candidate solutions.

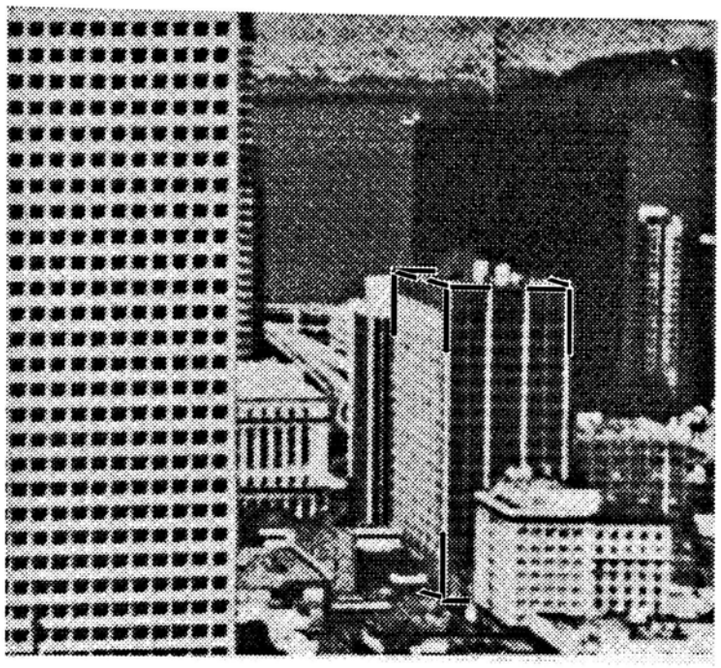


Figure 1.2: The image of a building of a simple rectangular structure already contains four OTVs whose branches are all visible.

- Developed a new algorithm to recover the orientation between two images by making use of the three branches of an OTV as an intermediate frame.
- Studied the performance of the new approach in comparison with existing approaches.
- Studied the performance of the new approach through real images.
- Analyzed the output of the new approach against input noise.

1.5 Organization of this Dissertation

In chapter 2 we review previous work on the problem. In chapter 3 we explain the theory of our approach, and we present a closed-form solution to estimate the extrinsic parameters of a stereo system. Some experimental results with simulated and real image data are given in chapter 4. Error Analysis is given in chapter 5. At the end we present a summary of the work in chapter 6. Part of the work is reported in [7, 8].

Chapter 2

Previous Work

Here we give a brief survey of previous work on the calibration problem. Extensive work has been done on the calibration problem. A comprehensive survey can be found in [28]. There are basically two approaches to tackle the problem. One is first to use some premeasured 3-D information to estimate the absolute orientation of each camera with respect to a world coordinate frame, and then to recover the relative orientation between the cameras, namely the 3×3 rotation matrix R and the 3×1 translation vector $\vec{t}$, using the chain rule of spatial transformation. We call this the *absolute orientation approach.* The other is to compute the relative orientation between the cameras directly from some feature correspondences across the stereo pictures, which we call the *relative orientation approach.*

2.1 Absolution orientation approach

Yakimovsky and Cunningham [35] proposed a method using this approach. With a linear sensor array and a high-quality lens, it provides 3-D measurement resolution of 1/400 [1]. Tsai [28] presented a method for estimating both the intrinsic and extrinsic parameters of a camera. Though accurate, the method demands for a well machined object with very precise predefined feature points. Horn [15, 16] proposed two methods to estimate the absolution orientation. Other work on recovering absolute orientation includes [11, 1, 31].

Absolute orientation approach requires the use of an object in 3-D with exactly known dimensions, and it computes additional information like the

[1] $\pm$ 5mm at distances of about 2m

absolute orientations of the cameras with respect to an arbitrary world coordinate frame attached to the object. Notice that it is only the relative orientation between the cameras, not their orientations with respect to any particular coordinate frame, which is usually of interest. More importantly, the idea of absolute orientation approach is to have a stereo camera setup rigidly constructed and calibrated once, so that a large number of stereo images can be taken afterwards using the same setup without the need of going through the calibration process again. This may not be feasible in many applications, especially when the objects of interest are far away and therefore a long baseline is required, an example of which is aerial photogrammetry [6]. To take stereo pairs of aerial images, the usual practice is to mount a camera under a plane and let it fly and take pictures at locations wide apart. Each pair of pictures thus have a different camera configuration, and they have to be calibrated based upon the image features alone as very often the objects in the pictures are not of known dimensions.

2.2 Relative orientation approach

The relative orientation approach is to determine directly the relative transformation between the stereo camera coordinate frames without using prior measures in 3-D coordinates. As illustrated later in section 3.1, the transformation parameters are determinable up to the rotation R and the direction of translation $\hat{t}$ only, *i.e.*, the absolute magnitude of the translation is a degree of freedom undeterminable with two views.

Many have proposed to use point correspondences[2] across the stereo images. It requires only five of them at minimum to determine R and $\hat{t}$ [17]. Unfortunately, determining R and $\hat{t}$ from five point correspondences involves solving 22 simultaneous equations of the second order [17] or 5 simultaneous equations of the third order [27]. Using more than five point correspondences, not only the problem can be made simpler, the redundant information can also be used to reduce the effect of noise to the solution.

Longuet-Higgins [22] described an algorithm that requires solving 8 simultaneous linear equations only. Tsai and Huang [30] independently discovered the 8-point algorithm. However, the accuracy is sensitive to both the magnitude and the direction of the translation. Longuet-Higgins also derived the

[2]By point correspondences, we mean the pixels on the two images projected from the same point in 3D.

necessary and sufficient conditions for intrinsic [23][3] and extrinsic [24] degeneracy, under which the linear algorithm will fail. Zhuang *et al.* [37] further generalized it to include the case that $\vec{t} = \vec{0}$, which was not covered in [23], and named it the surface assumption. However, they had done noise-free simulation only to verify the algorithm. Neither real image result nor simulation result with noise was shown. Recently, Hartley [14] extended the work of Longuet-Higgins and suggested an algorithm for the point correspondence problem without the intrinsic parameters being known *a priori*.

Tsai and Huang [29] have shown that while the 8-point algorithm is computationally more efficient than the iterative search used in solving the nonlinear motion equation, it doesn't have a way to reduce the effect of noise in the point positions. If the error of point positions is less than 3%, the error of motion parameters can be significantly reduced with more point correspondences being used. They had demonstrated this with the result on a real stereo pair using 22 point correspondences. R was estimated within 5.29% error. Faugeras and Toscani [11] implemented a similar algorithm using Kalman Filtering, and the method can be used to recover both the absolute and relative orientations.

Weng *et al.* [33] also described a method that generalizes the 8-point algorithm so that more than eight point correspondences can be used to reduce solution error. They gave a comprehensive illustration of the generalized 8-point algorithm as well as a systematic report on the error analysis. The paper also shows one real image result in addition to a number of simulation results concerning various aspects of the problem. With 12 point correspondences, R can be found within 2% to 5% error; yet the error in estimating $\hat{t}$ is not stable, ranging from 5% to 13%.

Table 2.1 shows the performance of some of the methods. The recovery of the translation vector, in particular, is not stable. Using the method described in [30], the $\hat{t}$ error under 20 point correspondences can be three times worse than the one under 9 point correspondences.

Some [25, 21, 32] have also proposed to use more abstract features like line correspondences for calibration. However, since the projection planes going through two lines in two views always intersect regardless of the relative transformation between the views, such an approach requires at least 3 views to do the calibration. Mitiche *et al.* [25] made use of the invariance of angular configuration in the calibration. They derived formulae for 4 lines over 3

[3]Vertices of a cube are found to be a set of degenerate data no matter how they are placed relative to the two cameras.

views. No result was shown. Liu and Huang [21] used 6 lines from 3 views. They had done noise-free simulations only, which are aimed at examining the convergence range of their method in solving for the rotation components. Weng *et al.* [32] used 13 or more lines over 3 views. Redundancy in the data provides an overconstrained system of equations to combat noise. Table 2.2 shows the performance of their method.

Algorithm	Simulation Condition	Error		
Tsai & Huang [29]	*2.5% perturbation on projected points*	*8 corr.*	*20 corr.*	
	Rotation error:	38.7%	2.4%	
	Translation error:	103.6%	29.66%	
Tsai & Huang [30]	*0.1% perturbation on projected points*	*8 corr.*		
	Rotation error:	1.36%		
	Translation error:	0.51%		
	0.5% perturbation on projected points	*8 corr.*		
	Rotation error:	6.96%		
	Translation error:	20.66%		
	1% perturbation on projected points	*8 corr.*	*9 corr.*	*20 corr.*
	Rotation error:	14.32%	3.69%	0.83%
	Translation error:	53.97%	3.52%	10.09%
	2% perturbation on projected points	*8 corr.*		
	Rotation error:	30.28%		
	Translation error:	94.63%		
	3% perturbation on projected points	*8 corr.*	*20 corr.*	
	Rotation error:	47.1%	3.28%	
	Translation error:	101.8%	93.46%	
Weng *et al.* [33]	*Quantization to* 256×256 *array*	*8 corr.*	*12 corr.*	*20 corr.*
	Rotation error:	6.1%	2.1%	0.6%
	Translation error:	16.8%	6%	2.4%

Table 2.1: Performance of typical methods using point correspondences.

Algorithm	Simulation Condition	Error		
Weng *et al.* [32]	*Quantization to* 512×512 *array*	*13 corr.*	*18 corr.*	*30 corr.*
	Rotation error:	31%	3.5%	2%
	Translation error:	40.5%	5.5%	2%

Table 2.2: Performance of a typical method using line correspondences.

Chapter 3

Calibration using OTV correspondences

3.1 Problem Statement

As shown in Figure 3.1, let $\vec{L}_i$ (i = 1, 2, or 3) be the branches of an OTV in 3-D, $\vec{l}_{vi}$ be their projections in view v (v = 1 or 2), and $\vec{w}_v$ be the line of sight of the vertex from view v. The normal $\vec{n}_{vi}$ of the projection plane going through image branch $\vec{l}_{vi}$ is given by

$$\vec{n}_{vi} = \vec{w}_v \times \vec{l}_{vi}$$

Since $\vec{L}_i$ is at the intersection of the projection planes with normals $\vec{n}_{1i}$ and $\vec{n}_{2i}$ respectively, if R and $\vec{t}$ are the rotation matrix and translation vector between the stereo views, $\vec{L}_i$ can be computed as

$$\vec{L}_i = \vec{n}_{1i} \times R\vec{n}_{2i}$$

for all i = 1, 2, or 3.

Our relative orientation problem can therefore be defined as: given $\{\vec{l}_{vi} : i = 1,2,3\}$ and $\{\vec{w}_v : v = 1,2\}$ for each OTV correspondence, find R and $\vec{t}$ such that the following constraints are satisfied:

1. *Epipolar Constraint*:

   Since the lines of sight of the vertex from the two views have to intersect in 3-D, the lines of sight and the translation vector have to be coplanar, *i.e.*,
   the scalar triple product

   $$(\vec{w}_1 \; R\vec{w}_2 \; \hat{t}) = 0$$

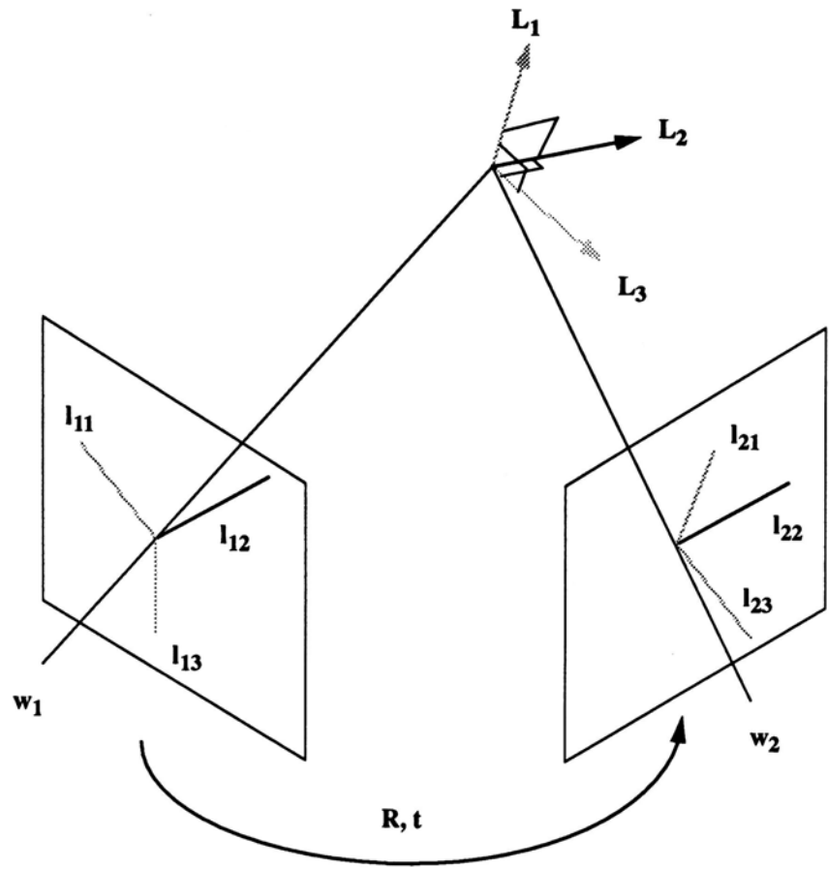


Figure 3.1: Stereo image calibration using OTV correspondences.

2. *Orthogonality Constraint*:

   Since $\{\vec{L}_i\}$ are mutually orthogonal,
   $\forall i \neq j$,

   $$(\vec{n}_{1i} \times R\vec{n}_{2i}) \perp (\vec{n}_{1j} \times R\vec{n}_{2j})$$

   where $\vec{n}_{vi} = \vec{w}_v \times \vec{l}_{vi}$.

Obviously the orthogonality constraint would allow R to be recovered, whereas the epipolar constraint would allow $\vec{t}$ to be recovered. It can also be observed that if $(R, \vec{t})$ satisfies the above constraints, so does $(R, k\vec{t})$ for any k. The absolute magnitude of the translation is therefore not determinable with two views, and the solution can be estimated in terms of a rotation matrix R and a translation direction $\hat{t}$ (a unit vector) only. However, many applications do not require the absolute magnitude of the translation. An example is the estimation of epipolar lines across the images for stereo matching, which we

show in appendix B. It is also simple to determine the remaining degree of freedom once any single piece of 3-D information about the scene is available.

An obvious approach to the problem is to compute R using a least-squares method so that the sum over a number of OTV correspondences:

$$\sum\sum_{i\neq j} ||(\vec{n}_{1i} \times R\vec{n}_{2i}) \cdot (\vec{n}_{1j} \times R\vec{n}_{2j})||^2$$

is minimized. Similar method can be used to compute $\vec{t}$ from the epipolar constraint. However, such an approach requires solving a set of nonlinear equations for the elements of R, and is discarded.

An interesting observation is, the three branches of each OTV, $\{\vec{L}_i\}$, themselves form an orthogonal coordinate frame in space, which we call frame 0 here. Moreover, it is found that $\{\vec{L}_i\}$ can be determined from the projection of the OTV in any single view (although there would be two possible solutions). As a result the rotation relationship ${}^vR_0$ between each image coordinate frame v (1 or 2) and frame 0 can be established. It therefore follows that we can use composite transformation to compute the rotation relationship between the stereo views, ${}^1R_2$, from ${}^1R_0$ and ${}^2R_0$, as if the OTV is an *intermediate coordinate frame*. The minimum number of OTV correspondences needed to have a unique solution of ${}^1R_2$ is found to be two. The translation vector $\vec{t}$ can be determined afterwards using the image coordinates of the vertex projections in the two views.

We will first describe how the orientation of an OTV in 3-D can be estimated from a single view. We then describe how we determine the rotation matrix and then the translation vector between stereo views, using two OTV correspondences across the images. We will also describe how the idea can be extended to use more than two OTV correspondences.

3.2 Recovering the orientation of an OTV from a single view

The problem is to compute the branches $\{\vec{L}_i\}$ of an OTV given their projections $\{\vec{l}_i\}$ in a view and the line of sight $\vec{w}$ of the vertex.

As *lines*, $\{\vec{L}_i\}$ have to lie on the respective projection planes and be orthogonal to one another, *i.e.,* they have to satisfy the *orthogonality constraints*:

$$\vec{L}_1 \cdot (\vec{w} \times \vec{l}_1) \;=\; 0 \tag{3.1}$$

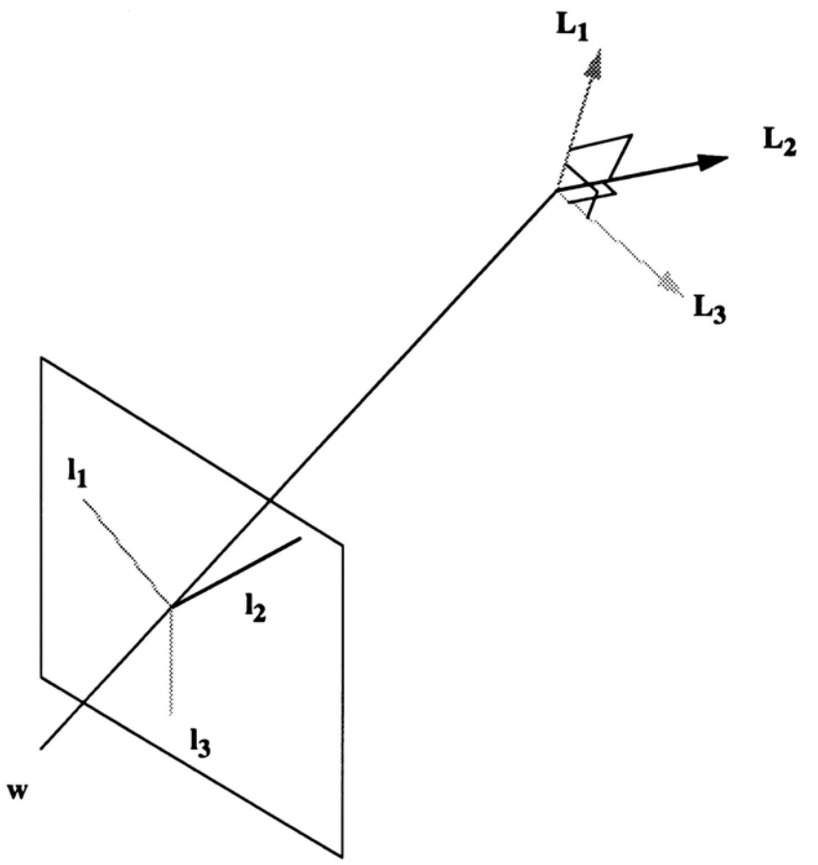


Figure 3.2: Projection of an OTV.

$$\vec{L}_2 = \pm\vec{L}_1 \times (\vec{w} \times \vec{l}_2) \tag{3.2}$$

$$\vec{L}_3 = \pm\vec{L}_1 \times \vec{L}_2 \tag{3.3}$$

$$\vec{L}_3 \cdot (\vec{w} \times \vec{l}_3) = 0 \tag{3.4}$$

It can be observed that if $\vec{L}_i$ satisfies equations (3.1)-(3.4), so does $-\vec{L}_i$. This leads to a high degree of ambiguity in the solution. Fortunately, there is another constraint for $\{\vec{L}_i\}$. As *vectors*, $\{\vec{L}_i\}$ have to be in the directions such that they project to the image vectors $\{\vec{l}_i\}$ but not $\{-\vec{l}_i\}$, *i.e.,* they have to satisfy the following what we call the *vector projection constraint*: For all $i = 1, 2, \text{or } 3$,

$$(w \times \vec{L}_i) \cdot (w \times \vec{l}_i) > 0 \tag{3.5}$$

The constraint would allow the sign of each $\vec{L}_i$ solution to be determined.

Since only the directions of $\{\vec{L}_i\}$ are determinable (and they cannot be zero vectors), we can assume unit vectors $\{\hat{L}_i\}$. Putting all the orthogonality

constraints together, there are 3 equations to solve for the 3 components of $\hat{L}_1$:

$$\hat{L}_1 \cdot (\vec{w} \times \vec{l}_1) = 0 \tag{3.6}$$

and

$$\|\hat{L}_1\|^2 = 1 \tag{3.7}$$

and

$$\begin{aligned} & \hat{L}_1 \times [\hat{L}_1 \times (\vec{w} \times \vec{l}_2)] \cdot (\vec{w} \times \vec{l}_3) && = 0 \\ \text{or} \quad & \{[\hat{L}_1 \cdot (\vec{w} \times \vec{l}_2)]\hat{L}_1 - \|\hat{L}_1\|^2(\vec{w} \times \vec{l}_2)\} \cdot (\vec{w} \times \vec{l}_3) && = 0 \\ \text{or} \quad & [\hat{L}_1 \cdot (\vec{w} \times \vec{l}_2)][\hat{L}_1 \cdot (\vec{w} \times \vec{l}_3)] && = (\vec{w} \times \vec{l}_2) \cdot (\vec{w} \times \vec{l}_3) \end{aligned} \tag{3.8}$$

These can be reduced to a single quadratic equation for the square of one of the unknowns, say that of the X-component, ${L_{1x}}^2$:

$$a({L_{1x}}^2)^2 + b({L_{1x}}^2) + c = 0 \tag{3.9}$$

where

$$\begin{aligned} a \;=\; & (4A^2B^2/C^4)\Big((2ABCDG - A^2CEG - C^3EG - A^2BFG + BC^2FG \\ & -A^2CDH - C^3DH + A^3FH + AC^2FH - A^2BDI + BC^2DI + A^3EI \\ & +AC^2EI - 2ABCFI) \\ & /(B^2CEG + C^3EG - B^3FG - BC^2FG + B^2CDH + C^3DH - 2ABCEH \\ & +AB^2FH - AC^2FH - B^3DI - BC^2DI + AB^2EI - AC^2EI + 2ABCFI)\Big) \\ & -\Big(1 + A^2/C^2 + (1 + B^2/C^2) \\ & (2ABCDG - A^2CEG - C^3EG - A^2BFG + BC^2FG - A^2CDH \\ & -C^3DH + A^3FH + AC^2FH - A^2BDI + BC^2DI + A^3EI + AC^2EI \\ & -2ABCFI) \\ & /(B^2CEG + C^3EG - B^3FG - BC^2FG + B^2CDH + C^3DH - 2ABCEH \\ & +AB^2FH - AC^2FH - B^3DI - BC^2DI + AB^2EI - AC^2EI + 2ABCFI)\Big)^2 \\ b \;=\; & (4A^2B^2/C^4)(C^3EG - BC^2FG + C^3DH - AC^2FH - BC^2DI \\ & -AC^2EI + 2ABCFI - 2ABCJ) \\ & /(B^2CEG + C^3EG - B^3FG - BC^2FG + B^2CDH + C^3DH - 2ABCEH \\ & +AB^2FH - AC^2FH - B^3DI - BC^2DI + AB^2EI - AC^2EI + 2ABCFI) \end{aligned}$$

$$
\begin{aligned}
& -2\Big(1 + A^2/C^2 + (1 + B^2/C^2) \\
& (2ABCDG - A^2CEG - C^3EG - A^2BFG + BC^2FG - A^2CDH - C^3DH \\
& +A^3FH + AC^2FH - A^2BDI + BC^2DI + A^3EI + AC^2EI - 2ABCFI) \\
& /(B^2CEG + C^3EG - B^3FG - BC^2FG + B^2CDH + C^3DH - 2ABCEH \\
& +AB^2FH - AC^2FH - B^3DI - BC^2DI + AB^2EI - AC^2EI + 2ABCFI)) \\
& \Big((1 + B^2/C^2)\,(C^3EG - BC^2FG + C^3DH - AC^2FH - BC^2DI \\
& -AC^2EI + 2ABCFI - 2ABCJ)/(B^2CEG + C^3EG - B^3FG \\
& -BC^2FG + B^2CDH + C^3DH - 2ABCEH + AB^2FH - AC^2FH \\
& -B^3DI - BC^2DI + AB^2EI - AC^2EI + 2ABCFI) - 1\Big)
\end{aligned}
$$

$$
\begin{aligned}
c = & -\Big((1 + B^2/C^2)\,(C^3EG - BC^2FG + C^3DH - AC^2FH - BC^2DI - AC^2EI \\
& +2ABCFI - 2ABCJ)/(B^2CEG + C^3EG - B^3FG - BC^2FG + B^2CDH \\
& +C^3DH - 2ABCEH + AB^2FH - AC^2FH - B^3DI - BC^2DI \\
& +AB^2EI - AC^2EI + 2ABCFI) - 1\Big)^2
\end{aligned}
$$

$$
\begin{aligned}
A &= (\hat{w} \times \hat{l}_1) \cdot \hat{e}_x \\
B &= (\hat{w} \times \hat{l}_1) \cdot \hat{e}_y \\
C &= (\hat{w} \times \hat{l}_1) \cdot \hat{e}_z \\
D &= (\hat{w} \times \hat{l}_2) \cdot \hat{e}_x \\
E &= (\hat{w} \times \hat{l}_2) \cdot \hat{e}_y \\
F &= (\hat{w} \times \hat{l}_2) \cdot \hat{e}_z \\
G &= (\hat{w} \times \hat{l}_3) \cdot \hat{e}_x \\
H &= (\hat{w} \times \hat{l}_3) \cdot \hat{e}_y \\
I &= (\hat{w} \times \hat{l}_3) \cdot \hat{e}_z \\
J &= (\hat{w} \times \hat{l}_2) \cdot (\hat{w} \times \hat{l}_3)
\end{aligned}
$$

and $\hat{e}_x$, $\hat{e}_y$, $\hat{e}_z$ are the unit vectors $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ respectively.

The equation gives four values for L_{1x}. The other two components L_{1y} and L_{1z} can be computed uniquely using any value of L_{1x}. In other words, there are generally four solutions for $\hat{L}_1$ from the orthogonality constraints alone. All four solutions are on the surface of a unit sphere centered at the origin in the X-Y-Z space, with two of them radially opposite to the others.

As mentioned above, the vector projection constraint would leave only two of the solutions of $\hat{L}_1$ valid. Once $\hat{L}_1$ is determined, $\hat{L}_2$ and $\hat{L}_3$ are readily obtainable from equations (3.2) and (3.3). Again, their signs can be determined using the vector projection constraint.

To summarize, there are two possible sets of solution for $\{\hat{L}_i\}$ from a single view. They are actually mirror reflections of each other about the plane orthogonal to the line of sight of the vertex.

If the vertex appears to be an A-junction (but not Y-junction), and if we can assume that the vertex is composed of opaque surfaces (which is usually the case), we can further remove the remaining ambiguity in the orientation of the OTV using the following what we call the *A-junction Opacity Constraint*: For a trihedral vertex (not necessarily an OTV) consisting of opaque surfaces to be visible as an A-junction,

$$\left[\hat{L}_1 \cdot (\hat{L}_2 \times \hat{L}_3)\right]\left[\vec{w} \cdot (\hat{L}_2 \times \hat{L}_3)\right] < 0$$

where $\hat{L}_1$, $\hat{L}_2$, $\hat{L}_3$ are the 3-D directional vectors of the middle branch and the other two branches, and $\vec{w}$ is the line of sight of the vertex from the image to the vertex point. In particular, if the vertex is an OTV, the condition can be reduced to

$$\hat{L}_1 \cdot \vec{w} < 0$$

The constraint is simple to explain. $\hat{L}_2$ and $\hat{L}_3$ form a plane with a normal vector $\hat{n}_{23} = \hat{L}_2 \times \hat{L}_3$. The plane divides the 3-D space into two halves regarding directions, one with $\vec{v} \cdot \hat{n}_{23}$ positive and the other negative for any directional vector $\vec{v}$. For the middle branch $\hat{L}_1$ of an A-junction to be visible, $\hat{L}_1 \cdot \hat{n}_{23}$ and $(-\vec{w}) \cdot \hat{n}_{23}$ must be of the same sign, *i.e.*, $(\hat{L}_1 \cdot \hat{n}_{23})(\vec{w} \cdot \hat{n}_{23}) < 0$.

If the vertex is an OTV, $\hat{n}_{23} = k\hat{L}_1$ for a $k = 1$ or -1. On substitution $(\hat{L}_1 \cdot \hat{n}_{23})(\vec{w} \cdot \hat{n}_{23}) = k^2(\vec{w} \cdot \hat{L}_1)$. The opacity constraint can therefore be reduced to $\hat{L}_1 \cdot \vec{w} < 0$.

Unless otherwise stated, we always assume the objects visible in a scene are opaque and apply the A-junction opacity constraint to all A-junctions throughout this paper.

3.3 Recovering the transformation parameters between two views

The problem here is, given the projections of an OTV in two images, how do we compute the rotation matrix R and the translation direction $\hat{t}$ between the

views? Our approach is to use the orthogonality constraints of an OTV to recover R first, using the OTV as an intermediate coordinate frame, and to use the epipolar constraint to recover $\hat{t}$.

3.3.1 Recovering R

For any OTV visible in both views, using the method described in section 3.2, we can compute the orientation of the OTV, $\{\hat{L}_i\}$, with respect to the two views separately. In addition, we can pick an arrangement of the 3 vectors, with $\hat{L}_1$ as the $\hat{X}$-vector and the other 2 vectors as the $\hat{Y}$-vector and the $\hat{Z}$-vector, such that the vectors $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ form a right-handed coordinate system ($\hat{X} \times \hat{Y} = \hat{Z}$). With such a coordinate frame 0, the rotation matrix from the image coordinate frame v (1 or 2) to the OTV coordinate frame 0 is given by:

$$ {}^{v}R_0 = [\ \hat{X}\ \ \hat{Y}\ \ \hat{Z}\ ] $$

Using composite transformation, the rotation matrix between the stereo views is then simply

$$ {}^{1}R_2 = {}^{1}R_0\ {}^{2}R_0{}^{T} \tag{3.10} $$

Since the ${}^{v}R_0$ for each view generally has two possible solutions, there are altogether 4 possible combinations of ${}^{v}R_0$'s to recover ${}^{1}R_2$. Again, fortunately, there is a fact available: whatever orientations of the OTV recovered separately from the two views are, there should exist a spatial transformation that allows one orientation to overlap exactly with the other under the specified branch correspondences. This constraint, which we call *orientation consistency constraint* of the OTV, would allow 2 possible solutions of ${}^{1}R_2$ to be ruled out.

To check whether $\{\hat{L}_i\}$ computed separately from the two views are consistent or not, we can use the following test. Suppose $\{\hat{L}_{i1}\}$ and $\{\hat{L}_{i2}\}$ are the directions of the OTV branches computed from the two views separately, where $i = 1$, 2 or 3 is referring to the ith branch of the OTV. Compute

$$ C = [(\hat{L}_{11} \times \hat{L}_{21}) \cdot \hat{L}_{31}][(\hat{L}_{12} \times \hat{L}_{22}) \cdot \hat{L}_{32}] \tag{3.11} $$

If $C > 0$, then the OTV orientations recovered from the two views separately are consistent; otherwise, they are not.

With only one OTV correspondence across the stereo views, there are still two possible solutions left for the rotation matrix ${}^{1}R_2$ (hereafter denoted by R)

between the stereo views. At least one more OTV correspondence is needed to have a unique solution for R.

We use a simple method to compute R from two OTV correspondences. We pick two solutions for the rotation matrix, one from each OTV correspondence, such that the Euclidean norm of their difference is minimum among all the combinations. With this set of rotation matrices $\{S_1, S_2\}$, we find the orthonormal matrix $\bar{R}$ that best approximates them, using a method described in appendix A. R then takes the optimal value of $\bar{R}$.

3.3.2 Recovering $\vec{t}$

As mentioned above, the translation vector can only be determined up to a scaling factor with two views. Once R is known, the unit translation vector, $\hat{t}$, can be recovered using the epipolar constraint. Every point correspondence, for example, the correspondence of the vertex positions of an OTV across the stereo views, gives a linear equation

$$\hat{t} \cdot (\vec{w}_1 \times R\vec{w}_2) = 0 \tag{3.12}$$

for the elements of $\hat{t}$, where $\vec{w}_i$'s are the lines of projection of the two views through the point correspondence.

Therefore two OTV correspondences will be just enough to determine the two degrees of freedom of $\hat{t}$. However, the vertices of the two OTVs have to be on different epipolar planes for the equations to be independent.

3.3.3 Summary of all the steps

We give a summary of all the steps used in the proposed method here:

1. Determine the direction vectors of each OTV, $\{\hat{L}_i\}$, with respect to each view, using the orthogonality constraints, the vector projection constraint, and the A-junction opacity constraint.

2. Determine the rotational relationship between each view and the OTV coordinate frame from $\{\hat{L}_i\}$. Find the rotation matrix R between the stereo views using composite transformation. There would be altogether four possible solutions for R.

3. Remove two possible solutions of R using the orientation consistency constraint of the OTV.

4. Each OTV correspondence therefore supplies two possible solutions for R. With 2 OTV correspondences across the stereo views, pick two solutions, one from each OTV correspondence, such that the norm of their difference is minimum. With this set of R values $\{S_1, S_2\}$, find the orthonormal matrix $\bar{R}$ that minimizes $\|\bar{R} - \frac{S_1+S_2}{2}\|$. R then takes the optimal value of the orthonormal matrix $\bar{R}$.

5. Recover $\hat{t}$ using the epipolar constraints of two point correspondences, possibly those of the vertices of two OTVs.

3.3.4 Recovering R and $\vec{t}$ using more than 2 OTVs

It is important that a solution method is extensible to use more than the minimum input data set to reduce the effect of noise to the solution. If more than two OTV correspondences are available across the images, we would make use of all of them to compute R and $\hat{t}$.

Suppose n OTV correspondences, where $n > 2$, are available, each of which supplying two possible solutions S_{1r} and S_{2r} for R $(r = 1, \ldots, n)$. We anticipate that one of the solutions supplied by each OTV correspondence is close to the same rotation value – the true R. With that in mind, we would first pick n S_{ir}'s ($i = 1$ or 2), one from each OTV correspondence, such that compared with other combinations they are "closest" to one another in the rotation matrix space, with the measure of "closeness" yet to be defined. We then take the best approximation of this set of rotation values as the final R.

To extract the set of rotation values which form the smallest cluster in the rotation value space and to give the best approximation of them, a measure of "closeness" of rotation values is needed. The Euclidean norm of difference of 3×3 matrices is not a good measure, as it excludes the fact that a rotation matrix has to be orthonormal. We use a measure of "closeness" defined and described in details in appendix A. The basic idea is to represent each rotation value as a unit quaternion [13, 5], and the "closeness" of two rotation values is defined as the absolute magnitude of the dot product of the two corresponding unit quaternions. The definition also allows a least-squares approximation of a set of rotation values to be computed, and that is what we use to compute the final R.

For $\hat{t}$, once R is known, again we can use a least-squares method to compute it: find $\hat{t}$ of unity magnitude such that

$$\sum_{r=1}^{n} \left| \hat{t} \cdot (\vec{w}_{1r} \times R\vec{w}_{2r}) \right|^2$$

is minimum, where $\vec{w}_{1r}$ and $\vec{w}_{2r}$ are the lines of sight of the rth OTV correspondence in the two views. The problem can be converted into a standard form: find $\hat{t}$ of unity norm that minimizes $\|A\hat{t}\|$, where

$$A = \begin{bmatrix} (\vec{w}_{11} \times R\vec{w}_{21})^T \\ (\vec{w}_{12} \times R\vec{w}_{22})^T \\ \cdots \\ (\vec{w}_{1n} \times R\vec{w}_{2n})^T \end{bmatrix}$$

The optimal solution for $\hat{t}$ is the unit eigenvector of the matrix $A^T A$ associated with its smallest eigenvalue [26].

Chapter 4

Experimental Results

4.1 Simulated Data Experiments

We have carried out extensive simulation experiments to evaluate the performance of the proposed method. We have put in quantization noise, just like what the others [33] did, to examine the performance of the proposed method. Unless otherwise stated, in all the simulation experiments presented below, the image resolution is 256 by 256. In addition, error is defined in terms of the Euclidean norm of the difference of the computed value and the correct value:

$$R \text{ error} = \frac{\left\| R_{\text{computed}} - R_{\text{true}} \right\|}{\| R_{\text{true}} \|}$$

and

$$\hat{t} \text{ error} = \left\| \hat{t}_{\text{computed}} - \hat{t}_{\text{true}} \right\|$$

4.1.1 Error versus the smallest angle among the projected branches of an OTV

The case of having two image branches of an OTV being so close to each other that they overlap (their projection planes are coplanar) is obviously a degenerate case for the proposed algorithm. So is the case of having an image branch of zero length (the branch is parallel to the line of sight of the vertex). The near-degenerated cases are therefore that the projected branches have narrow angles among them, and that the branches are short. We have done some experiments to examine how sensitive the proposed algorithm is to the smallest angle among the projected branches and to the shortest length of the branches.

Figure 4.5 to 4.6 show the error in R versus the smallest angle among the projected branches of an OTV, when different lengths of the projected branches are visible. The OTV in the experiment is one with its vertex along the optical axis of the camera. It has its branches originally making equal angles with the image plane, and was rotated gradually about one of its branches as the experiment proceeded. Since one OTV correspondence provides two possible solutions for R, we assume additional OTV correspondences will resolve the ambiguity and we use the solution closer to the true R to calculate the error. The experiment shows that the error in computing R is not significantly affected by the length of the projected branches, as long as they are more than 15 pixels long. It can also be seen that as long as the smallest angle is greater than 10°, the R error can be as low as 2% with only one single OTV correspondence.

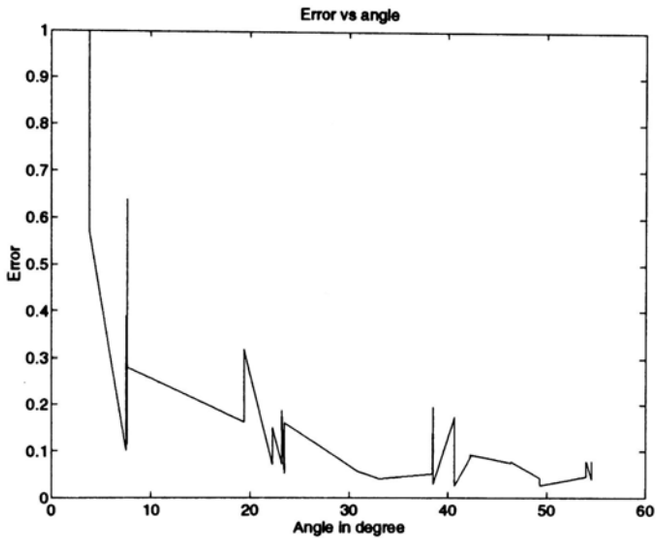


Figure 4.1: R error versus the smallest angle among the projected branches of an OTV (branch length = 15 pixels).

4.1.2 Comparison with a point correspondence algorithm

In a recent work, Weng *et al.* [33] have proposed a point correspondence method and have, among very few others in the calibration literature, presented extensive experimental results to illustrate the performance of their method. The study is rigorous and the results are impressive. We regard their method (hereafter the point correspondence method) as one representative

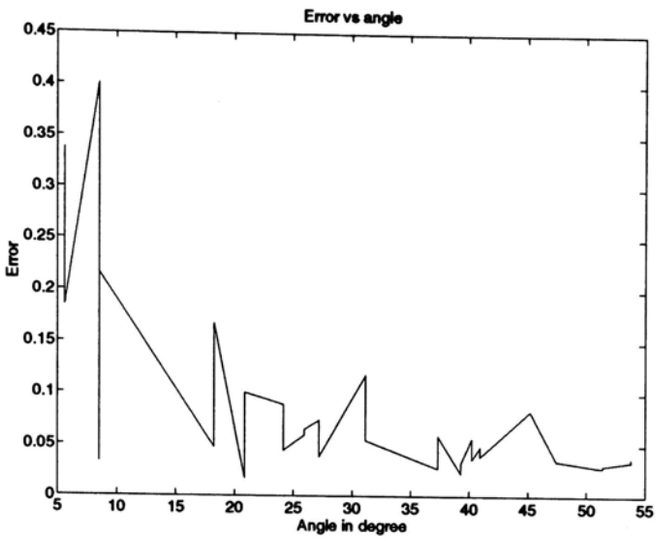


Figure 4.2: R error versus the smallest angle among the projected branches of an OTV (branch length = 20 pixels).

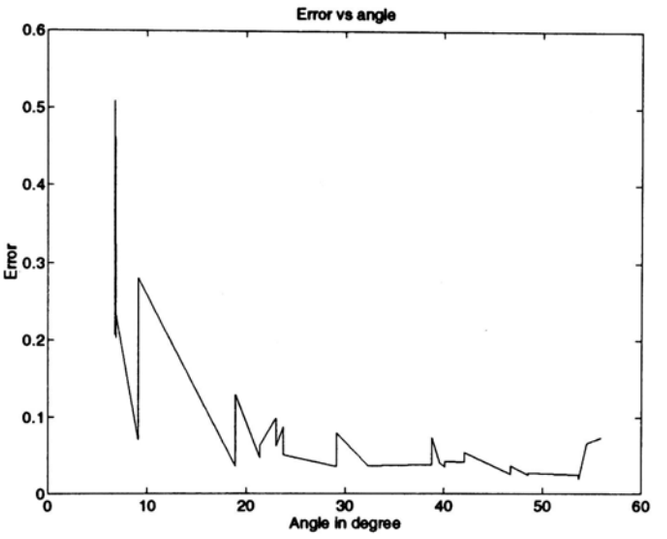


Figure 4.3: R error versus the smallest angle among the projected branches of an OTV (branch length = 25 pixels).

method using point correspondences, and we have done a simulation study of our method (hereafter the OTV correspondence method) under similar conditions to examine if there are major advantages of using angle constraints via OTV correspondences over position constraints via point correspondences.

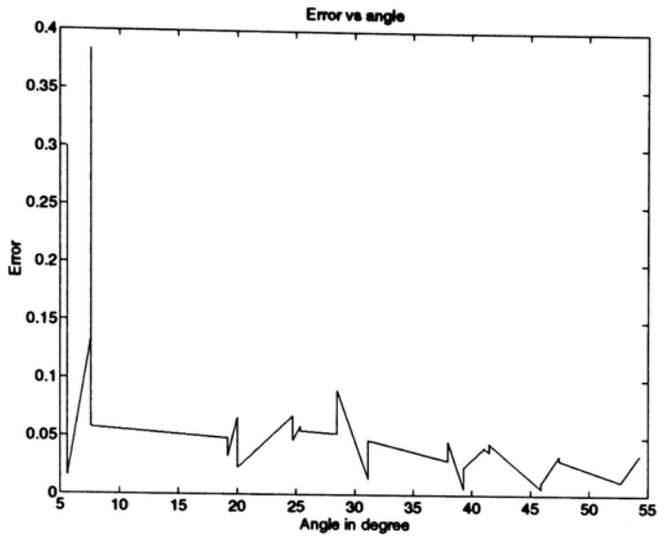


Figure 4.4: R error versus the smallest angle among the projected branches of an OTV (branch length = 30 pixels).

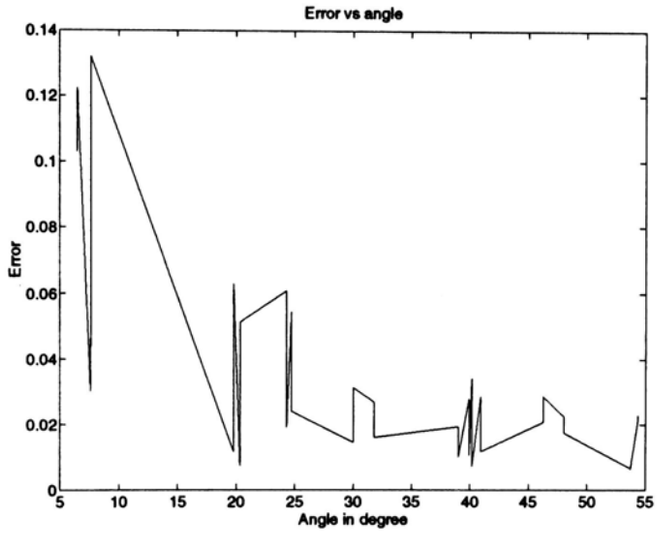


Figure 4.5: R error versus the smallest angle among the projected branches of an OTV (branch length = 45 pixels).

The simulation starts up with 6 randomly generated points in a $10 \times 10 \times 10$ cube. Two OTVs are then formed using the method stated in [15]. The center of this cube is defined as the object center, which is always 11 units away from the optical center of the camera. The OTVs (each composed of four points) then project perspectively to the camera to form images. The object distance

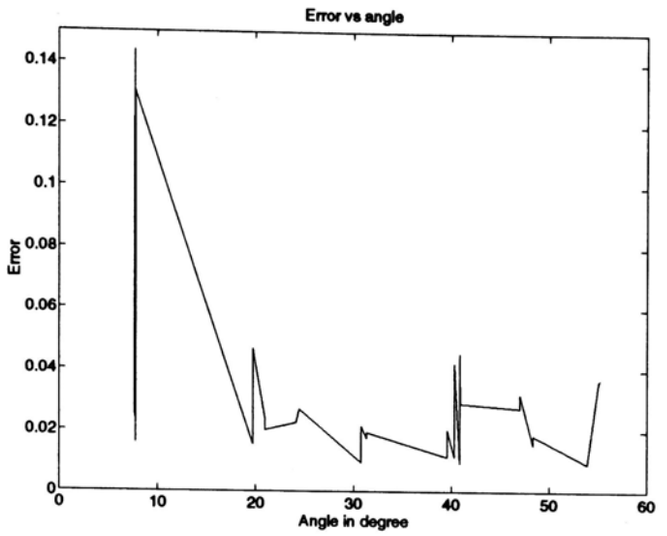


Figure 4.6: R error versus the smallest angle among the projected branches of an OTV (branch length = 60 pixels).

is therefore always 11 units. Image size is 2 unit with 128 pixels/unit. Focal length is 1 unit.

The following steps have been taken to ensure valid OTV projection:

- If a branch of an OTV intersects with the image plane, the branch would be cropped up to the point of intersection with the image plane.
- If there is an OTV whose vertex projection is out of the image, the image is rejected.
- If there is an angle among the projected branches of an OTV smaller than 15°, the image is rejected.

Figures below illustrate the performances of the OTV correspondence method and the point correspondence method under various conditions. The simulation setup is exactly the same as the one used in [33]. In particular, rotation is expressed as a rotation axis plus a rotation angle. Detail conditions for each simulation can be found beneath the corresponding figure. For example, the rotation R about an axis (1, -0.2, 0.2) by 8° is equivalent to

$$R = \begin{bmatrix} -0.9993 & 0.0286 & 0.025 \\ -0.025 & 0.9906 & -0.1343 \\ -0.0286 & 0.1336 & 0.9906 \end{bmatrix}$$

All the simulation results presented were averaged ones from 1000 trials of randomly generated point or OTV positions. In all the experiments, except the one of error versus the number of feature correspondences used, *12* point correspondences (it was shown in [33], and can be seen from Figure 4.7, that accuracy does not improve much with more than 12 point correspondences) were employed for the point correspondence method, and *2* OTV correspondences (the minimum number that allows the algorithm to work) were employed for the OTV correspondence method.

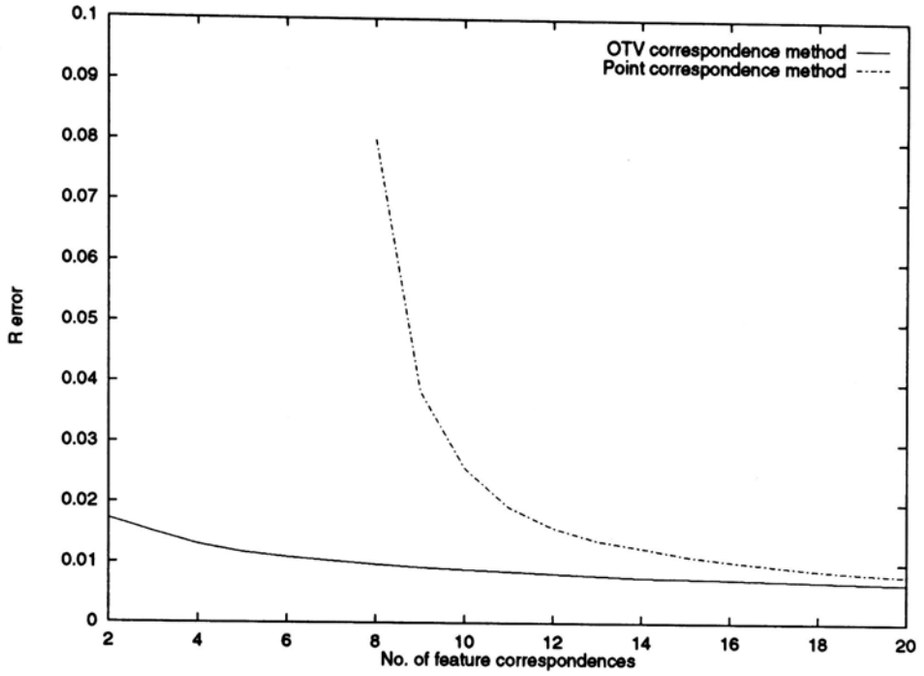
Figure 4.7 compares the performances of the point correspondence method and the OTV correspondence method when different number of feature correspondences are used. As expected, the more correspondences are used, the more accurate the solution is for both methods. However, it can be seen that the OTV correspondence method is more stable towards the number of feature correspondences used, and even with only 2 OTV correspondences, R error is already as small as 1.8%.

We do not intend to compare the performance of the two approaches under the same number of correspondences, as the features are not the same. Figure 4.7 just compares how the performances of the two methods improve with increasing number of feature correspondences.

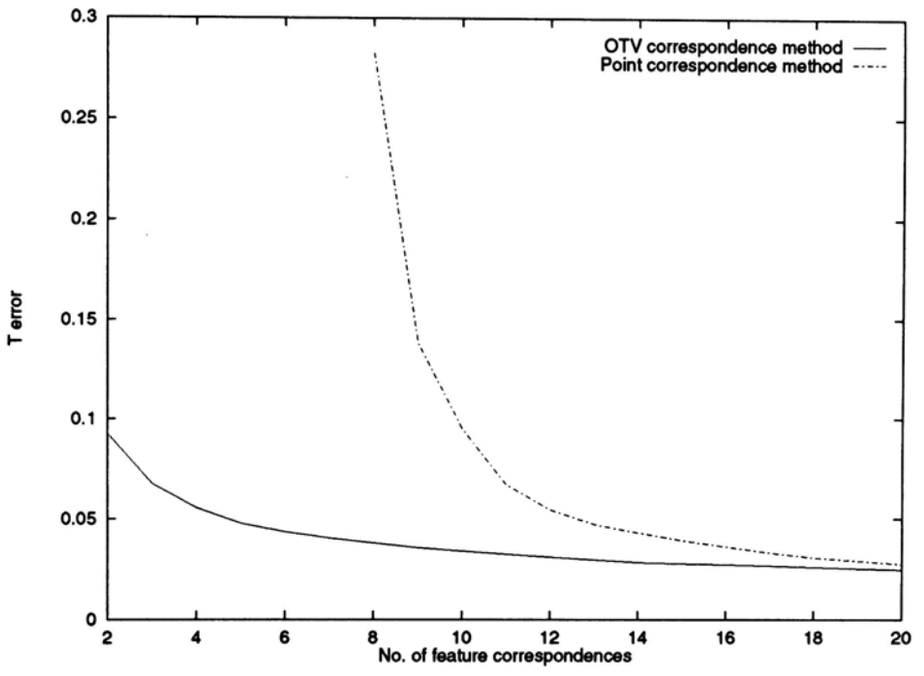
Figure 4.8 shows the simulation result of the point correspondence method when the direction of translation is varied. The figure also shows the simulation result of the OTV correspondence method under the same simulation condition. It can be seen that the OTV correspondence method is generally much more accurate and insensitive to the direction of the translation.

Figures 4.9, 4.10, and 4.11 are other sets of simulation results when the direction of translation is varied gradually from being orthogonal to the optical axes to being parallel to the optical axes. Again, it can be observed that the OTV correspondence method is generally less sensitive to the direction of translation, and is much more accurate in computing both R and $\hat{t}$. It is only under the extreme situation when the motion between the cameras is almost parallel to their optical axes, *i.e.,* one is right in front of another, that the error in $\hat{t}$ computed from the OTV correspondence method is larger. However, this is not an usual configuration in stereo analysis. On the other hand, the error in computing R using the OTV correspondence method is independent of the translation direction at all.

However, although $\hat{t}$ error using the OTV correspondence method is less dependent of the direction of translation, it is also generally larger (which



Rotation errors



Translation errors

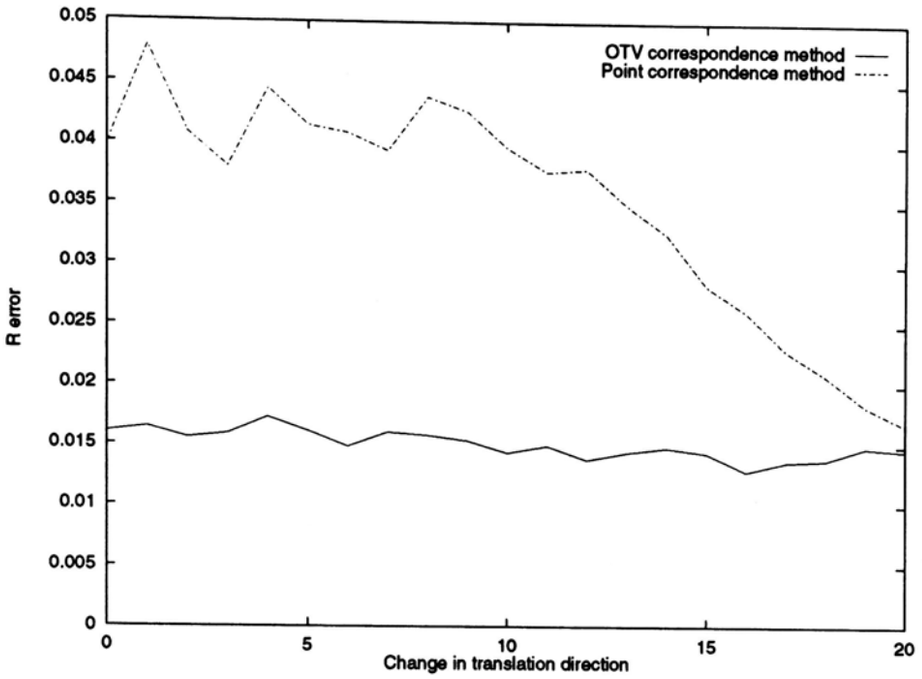
Figure 4.7: Error versus number of feature correspondences. Condition: rotation axis (0.9, 1, -0.8), rotation angle 5°, translation (-0.5, 0.5, 3).

is not surprising, as here we only use two point (vertex) correspondences to compute $\hat{t}$ from the epipolar constraint, while the other algorithm is using 12 of them).

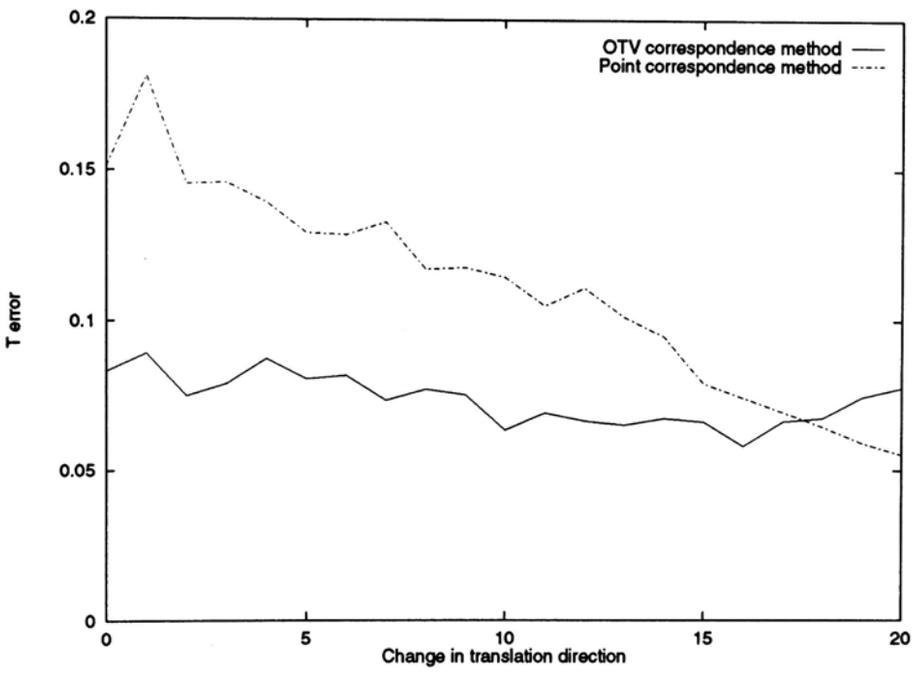
Figure 4.12 shows the simulation result when the magnitude of translation is varied. The $\hat{t}$ error for both methods generally increases when the magnitude of the translation decreases, which is logical as $\hat{t}$ error is defined to be the absolute error divided by the translation magnitude. However, the OTV approach is observed to be generally more accurate.

Figures 4.13, 4.14, and 4.15 show the simulation results when the magnitude of rotation angle is varied. It can be seen that both methods are generally independent of the amount of rotation, but the OTV correspondence method is observed to be much more accurate.

Figure 4.16 (with small rotation angle) and Figure 4.17 (with large rotation angle) show the simulation results when the rotation axis is varied. Both methods are observed to be independent of the axis of rotation, and again, the OTV correspondence method is generally much more accurate.

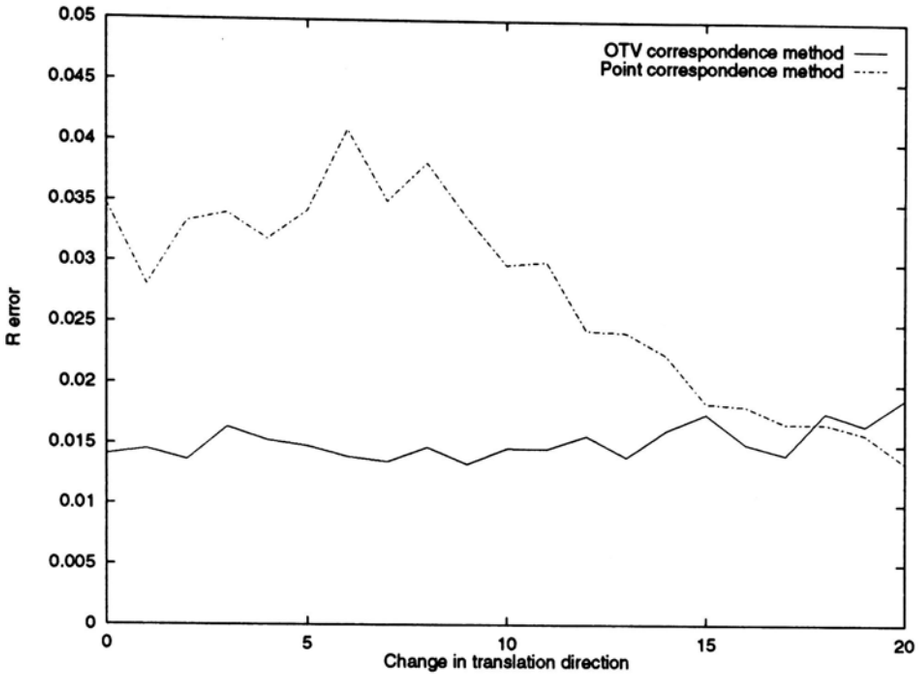


Rotation errors

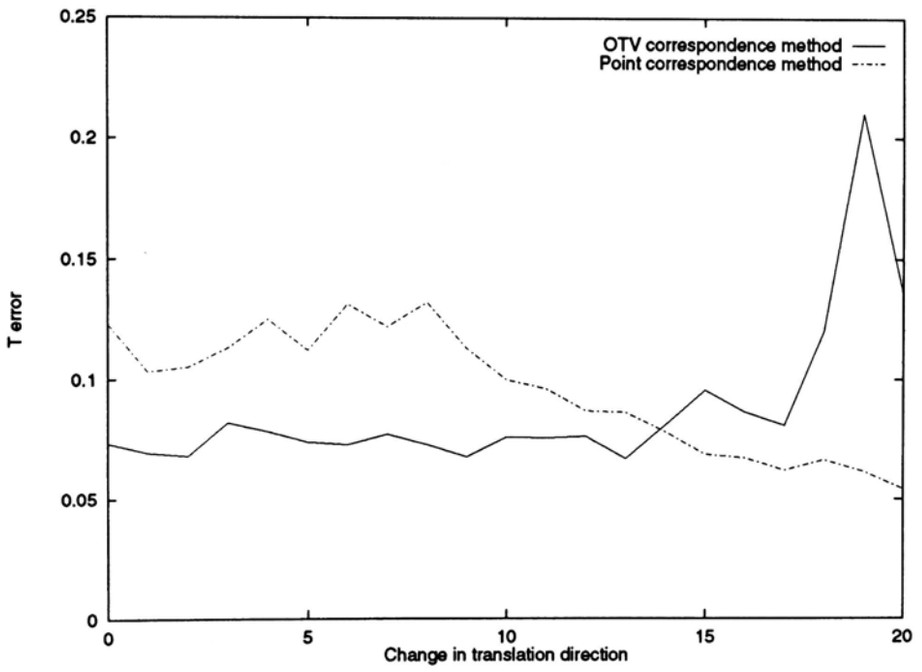


Translation errors

Figure 4.8: Error versus direction of translation. Condition: rotation axis (1, -0.2, -0.2), rotation angle 8°, translation from (1, 3, 0) to (1, 0, 3) evenly spaced 21 translation directions with magnitude 3.

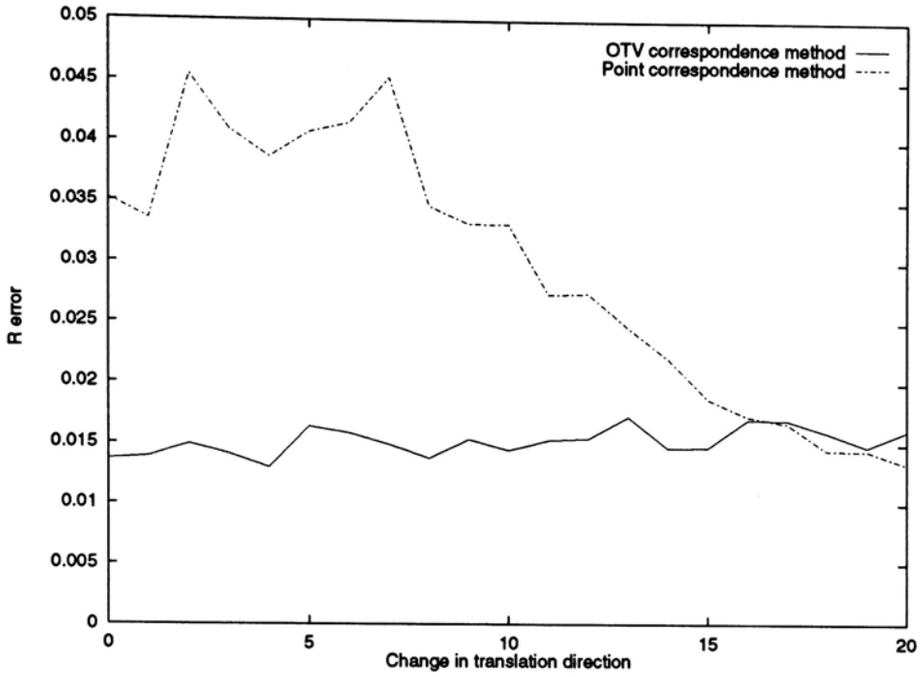


Rotation errors

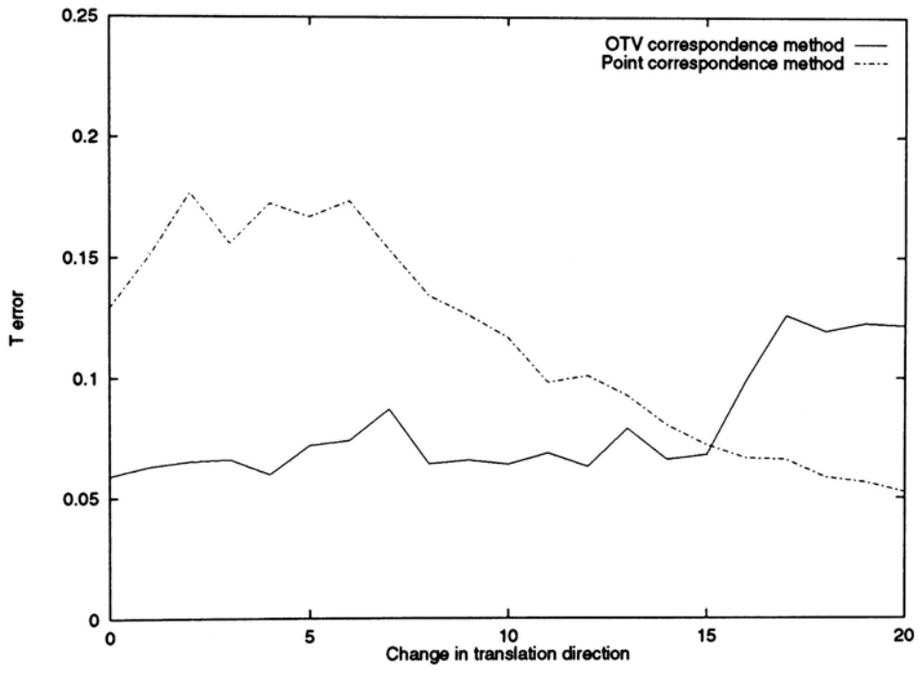


Translation errors

Figure 4.9: Error versus direction of translation. Condition: rotation axis (0, 1, 0), rotation angle 5°, translation from (0, 1, 0) to (0, 0, -1) evenly spaced 21 translation direction.

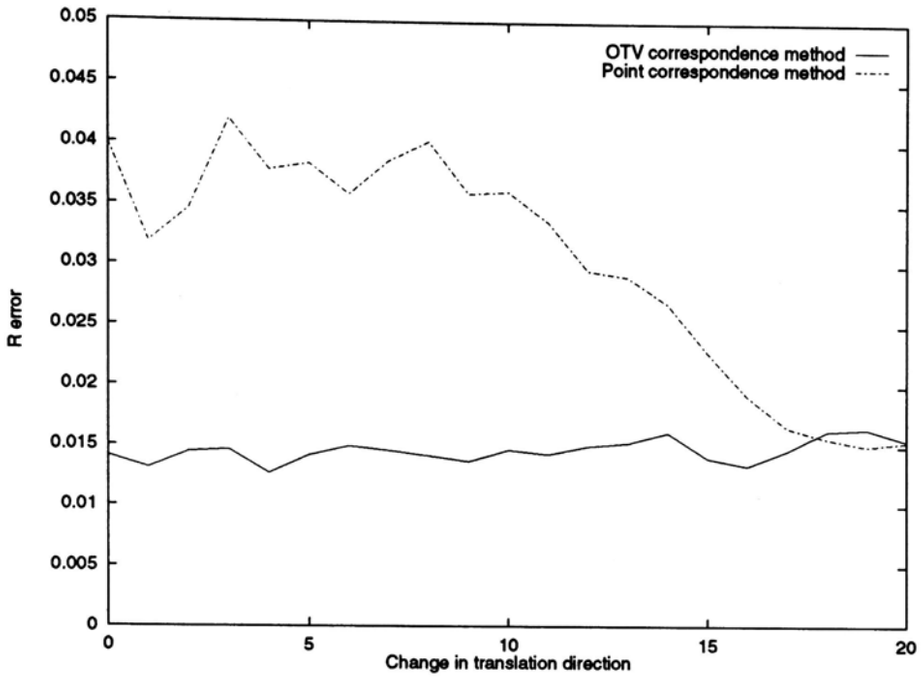


Rotation errors

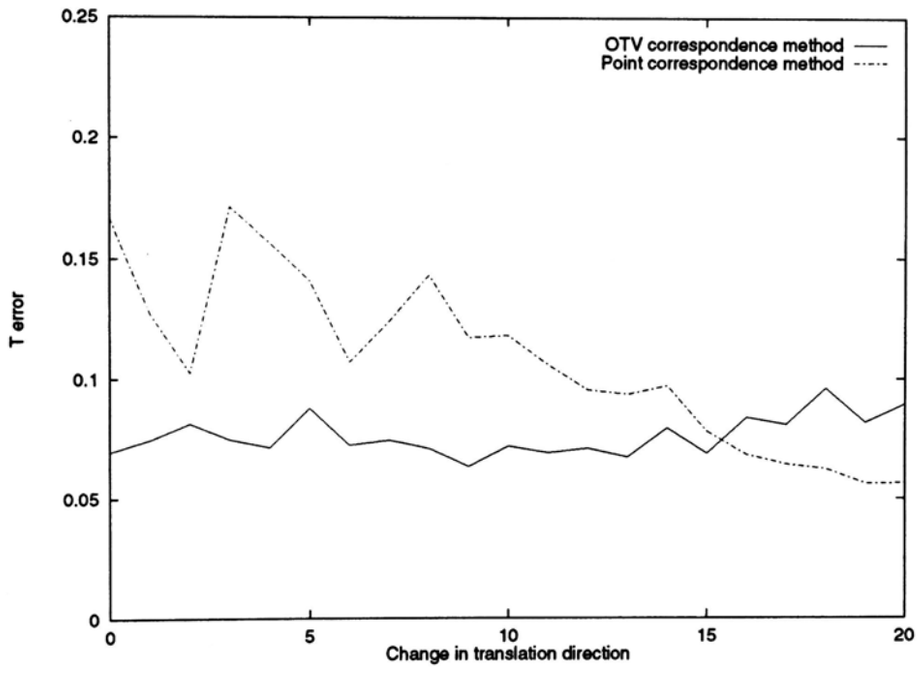


Translation errors

Figure 4.10: Error versus direction of translation. Condition: rotation axis (0, 1, 0), rotation angle 5°, translation from (1, 0, 0) to (0, 0, -1) evenly spaced 21 translation direction.

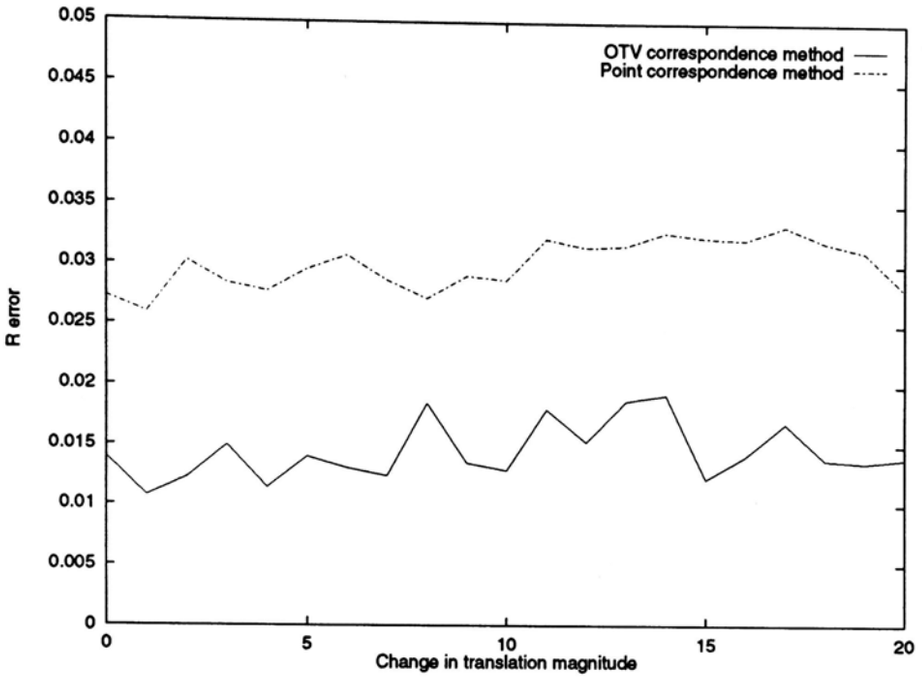


Rotation errors

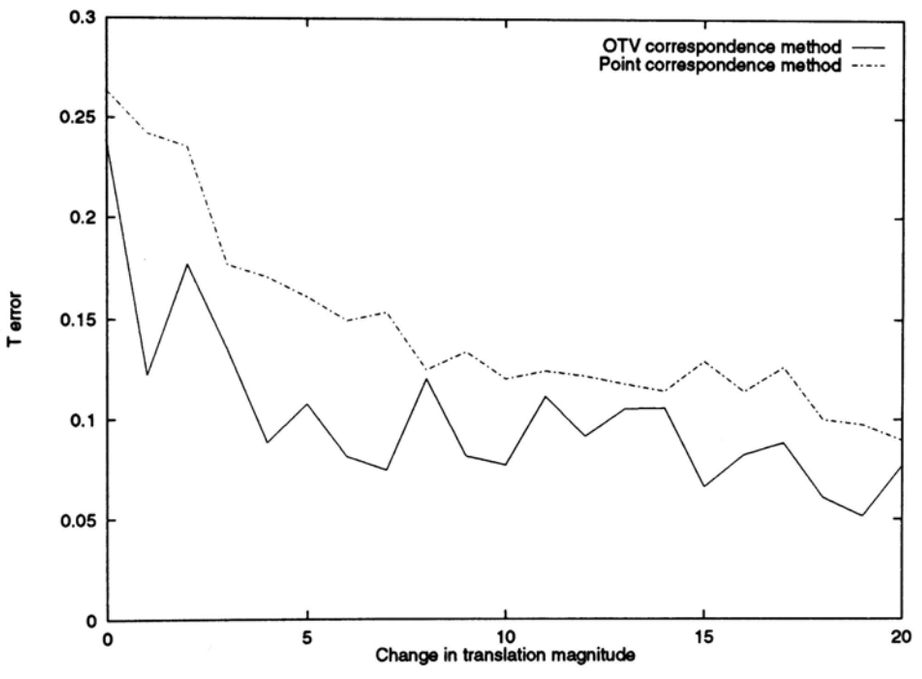


Translation errors

Figure 4.11: Error versus direction of translation. Condition: rotation axis (0, 1, 0), rotation angle 5°, translation from (0, 0, 1) to (0, 0, -1) evenly spaced 21 translation direction.

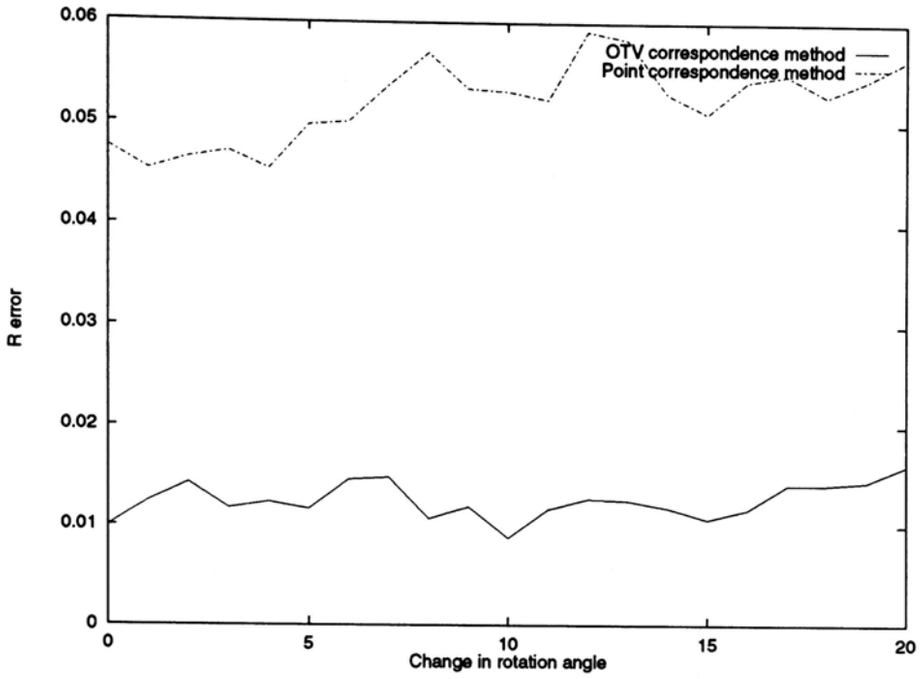


Rotation errors

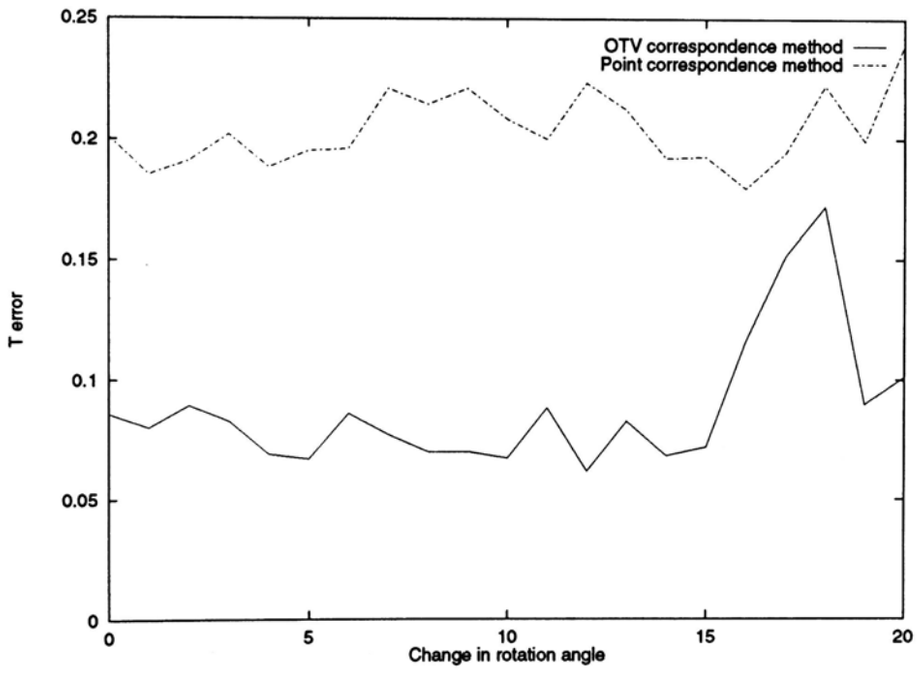


Translation errors

Figure 4.12: Error versus direction of translation. Condition: rotation axis (0, 1, 0), rotation angle 5°, translation from (0, 0.5, -0.5) to (0, 4.5, -4.5) evenly spaced 21 translation magnitudes.

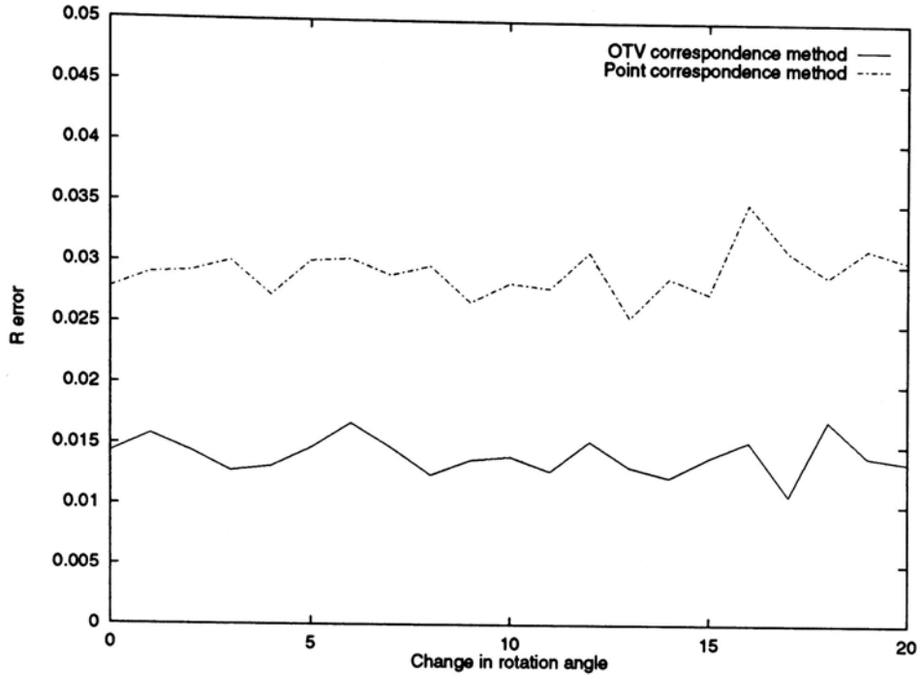


Rotation errors

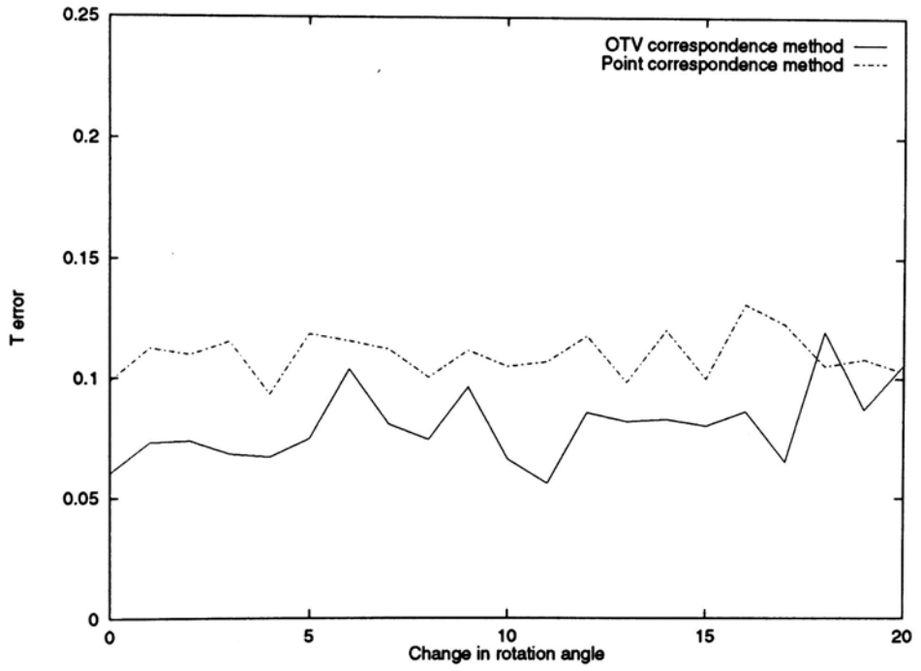


Translation errors

Figure 4.13: Error versus amount of rotation. Condition: rotation axis (0, 1, 0), from 0 to 30° evenly spaced 21 rotation angles, translation (0, 1.732, 1.732).

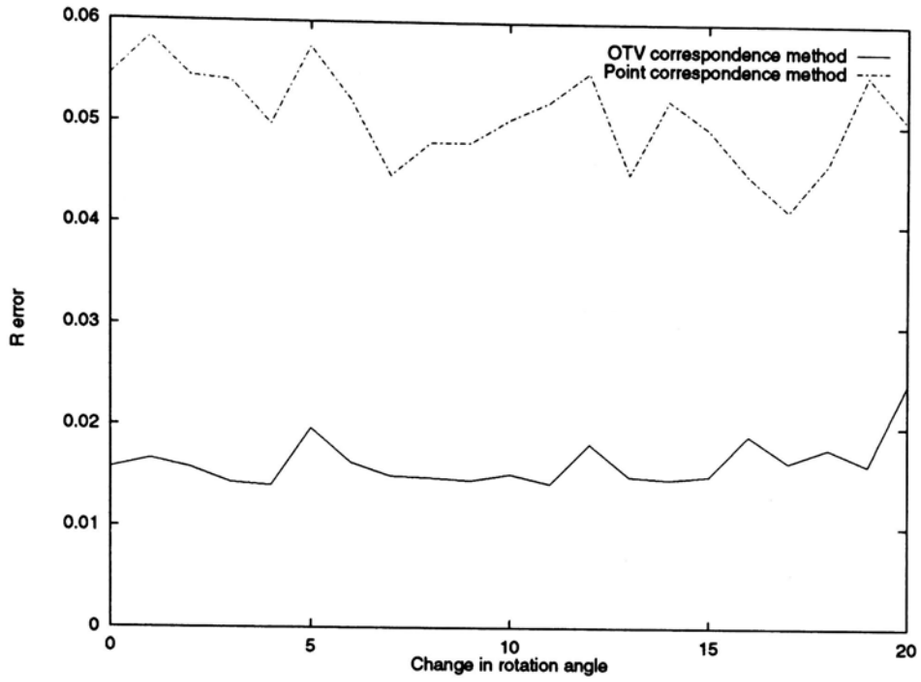


Rotation errors

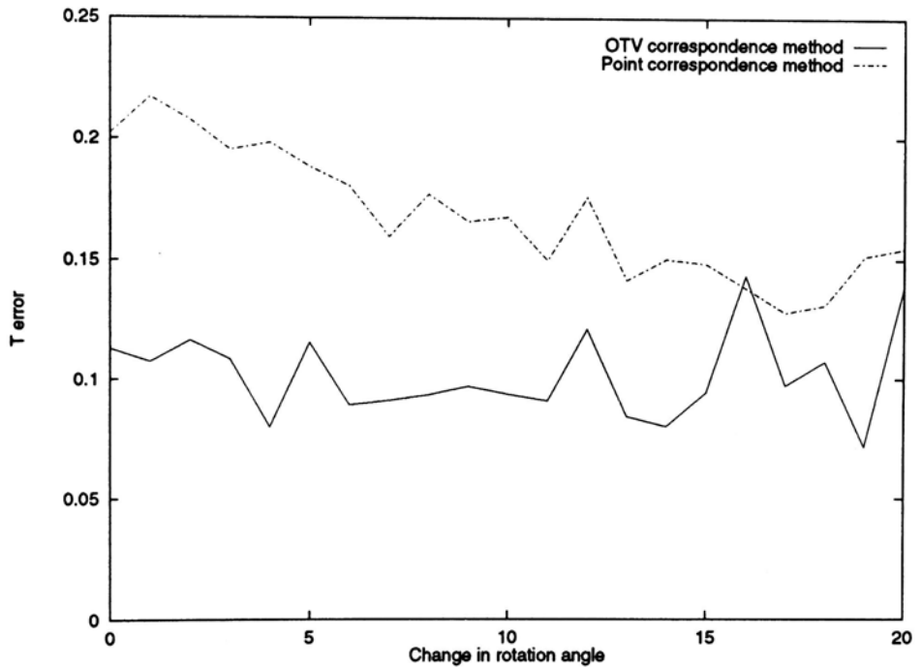


Translation errors

Figure 4.14: Error versus amount of rotation. Condition: rotation axis (0, 0, -1), from 0 to 30° evenly spaced 21 rotation angles, translation (0, 1.732, 1.732).

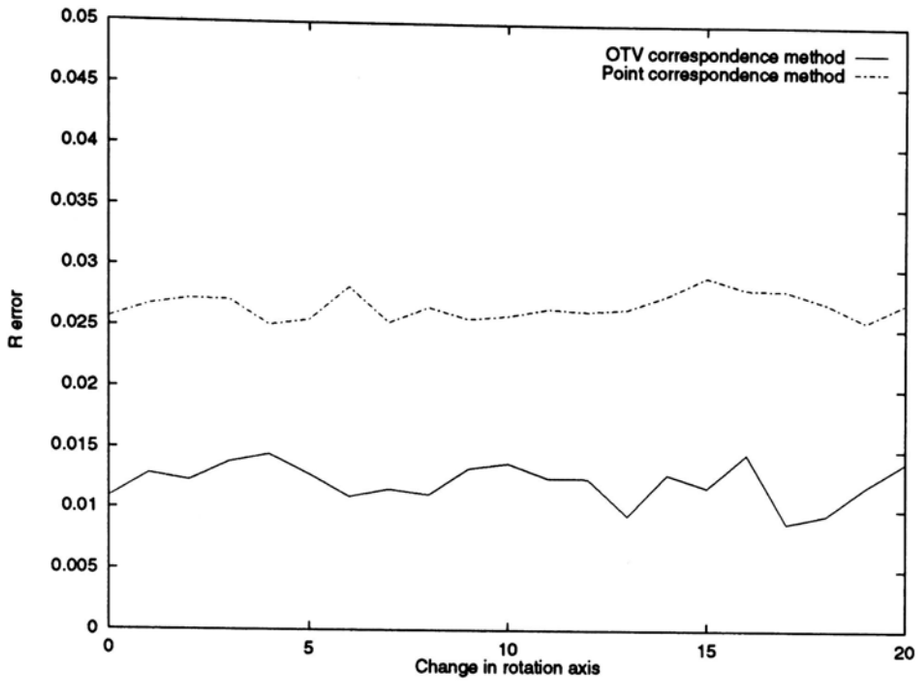


Rotation errors

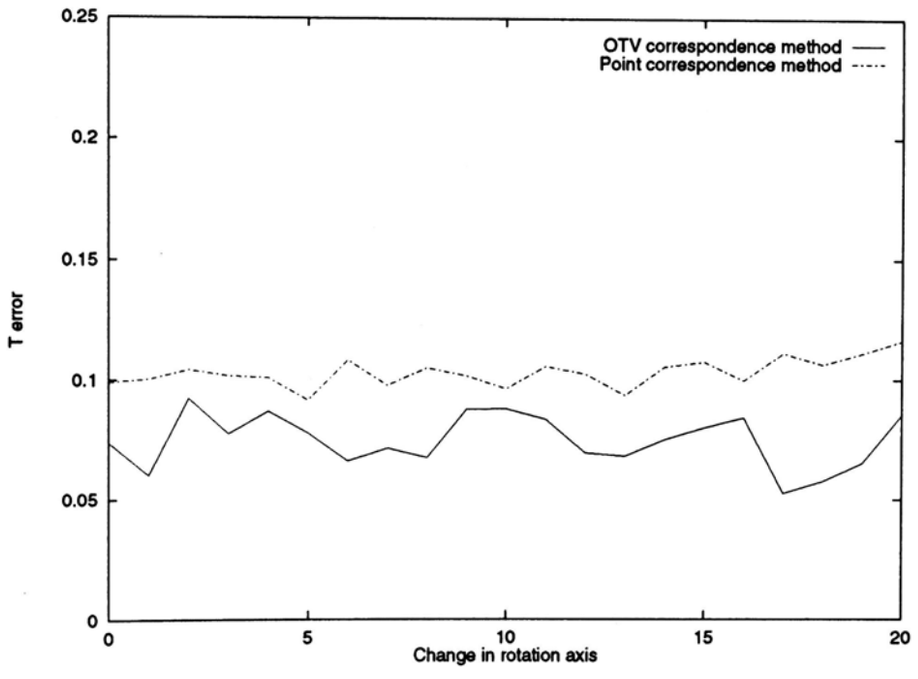


Translation errors

Figure 4.15: Error versus amount of rotation. Condition: rotation axis (1, 1, -1), from 0 to 30° evenly spaced 21 rotation angles, translation (1.732, 1.732, 1.732).

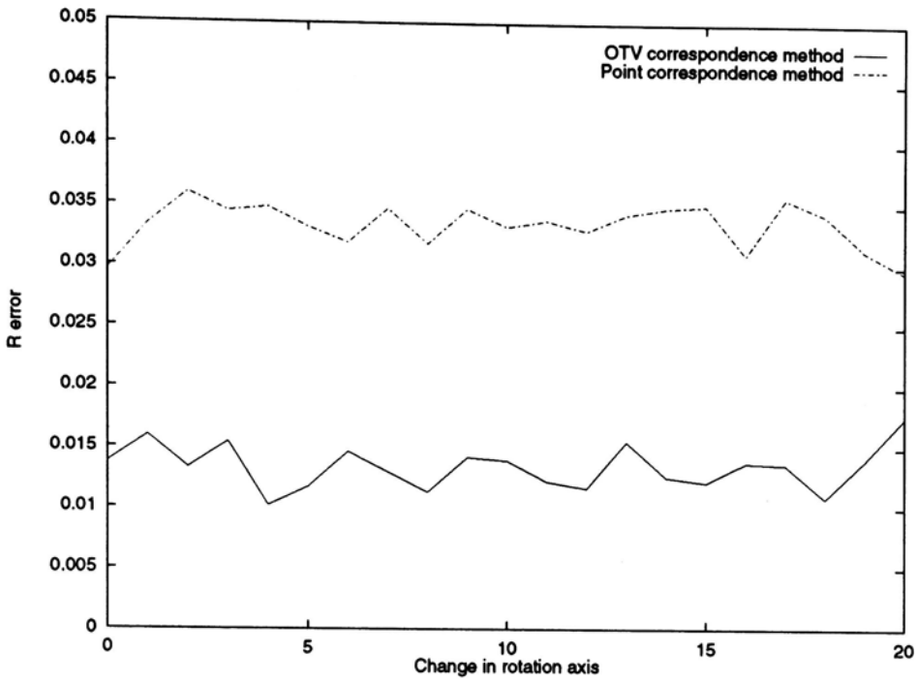


Rotation errors

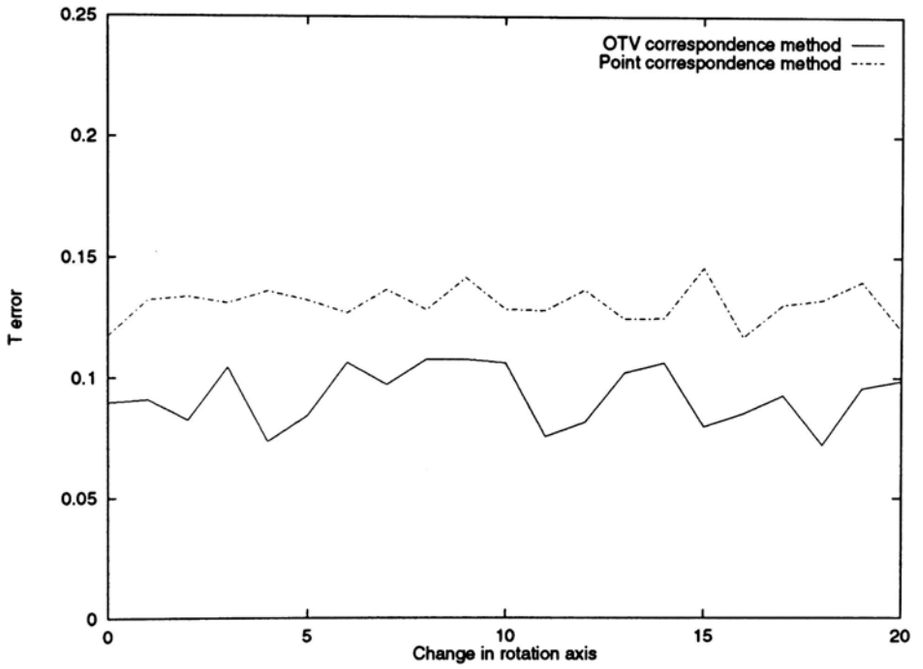


Translation errors

Figure 4.16: Error versus change in rotation axis. Condition: from (0, 1, 0) to (0, 0, -1) evenly spaced 21 rotation axes, angle 5°, translation (0, 1.732, 1.732).



Rotation errors



Translation errors

Figure 4.17: Error versus change in rotation axis. Condition: from (0, 1, 0) to (0, 0, -1) evenly spaced 21 rotation axes, angle 30°, translation (0, 1.732, 1.732).

4.2 Real Image Experiment

Since ground truth is not precisely known for real image data, it is difficult to evaluate the performance of an algorithm even if it does recover some parameters from the data. However, to get a feeling of how the proposed algorithm performs on real data, we tried our algorithm on some real stereo pairs to estimate R and $\hat{t}$ from the image features. To let the performance be examined visually, we computed the epipolar lines from the estimated R and $\hat{t}$ (it can be shown that the epipolar lines are independent of the absolute magnitude of the translation, as illustrated in appendix B), and plotted them on the original images. Corresponding points in the images should lie on corresponding epipolar lines for ideal results. An additional note is, although there have been quite some work on extracting corners from images [10, 19, 38, 36], at the time of performing the experiment we only determined the image coordinates of the OTV projections visually from the raw image data.

Figure 4.18 shows synthetic stereo images of an urban scene with streets and buildings (source: Harlyn Baker, Control Data Corporation) used in [2]. They are of dimensions 256 by 256. There is no rotation between the views, *i.e.*, $R = I$, and the translation between the views is along the horizontal axis, *i.e.*, $\hat{t} = [1\ 0\ 0]^T$. Using seven OTV correspondences circled in Figure 4.19, the proposed algorithm estimated that[1]

$$R = \begin{bmatrix} 0.9998 & -0.0008 & -0.0190 \\ 0.0007 & 1.0000 & -0.0040 \\ 0.0190 & 0.0040 & 0.9998 \end{bmatrix}$$

and

$$\hat{t} = \begin{bmatrix} 0.9998 \\ -0.0108 \\ 0.0188 \end{bmatrix}$$

which are very close to the true values. The epipolar lines of the stereo images were computed using the estimated R and $\hat{t}$ values, and they are shown in Figure 4.20. It can be seen that the epipolar lines are close to being horizontal, as they should be ideally.

Not being able to find any standard real image data in the calibration field for benchmarking purposes, we took some images ourselves for performance

[1] In this section, we present the R and $\hat{t}$ in a more intuitive manner, they represent the motion of the camera instead of the motion of the spatial vectors as was used in previous section.

study. Results on two stereo pairs, one indoor and another outdoor, are shown here. All were taken using the Canon Still Video Camera RC-560. The image resolution is 752 by 582, and the image size is $6.3 \times 4.8\text{mm}^2$. We used a focal length of 8 mm for the indoor scene and 24 mm for the outdoor scene.

Figure 4.21 shows two images of an indoor scene from different angles. It was estimated from eight OTV correspondences circled in Figure 4.22 that

$$R = \begin{bmatrix} 0.9615 & 0.1901 & -0.1985 \\ -0.1733 & 0.9799 & 0.0988 \\ 0.2133 & -0.0606 & 0.9751 \end{bmatrix}$$

and

$$\hat{t} = \begin{bmatrix} -0.8222 \\ 0.2921 \\ -0.4886 \end{bmatrix}$$

The epipolar lines were again computed using the estimated values of R and $\hat{t}$ and are shown in Figure 4.23. It can be seen that the result is satisfactory. Corresponding points in the images lie more or less on corresponding epipolar lines.

Figures 4.24, 4.25, and 4.26 show results on another stereo pair of images. The scene is an overview of Hong Kong. It was estimated from five OTV correspondences that

$$R = \begin{bmatrix} 0.9992 & -0.0373 & -0.0107 \\ 0.0371 & 0.9992 & -0.0178 \\ 0.0113 & 0.0174 & 0.9998 \end{bmatrix}$$

and

$$\hat{t} = \begin{bmatrix} 0.5071 \\ -0.2862 \\ -0.8130 \end{bmatrix}$$

The result is still satisfactory despite the weak contrast of the building boundaries and the hardly noticeable lengths of the OTV branches. We believe significantly better result can be obtained if the edges are better localized with help from sophisticated edge detectors.

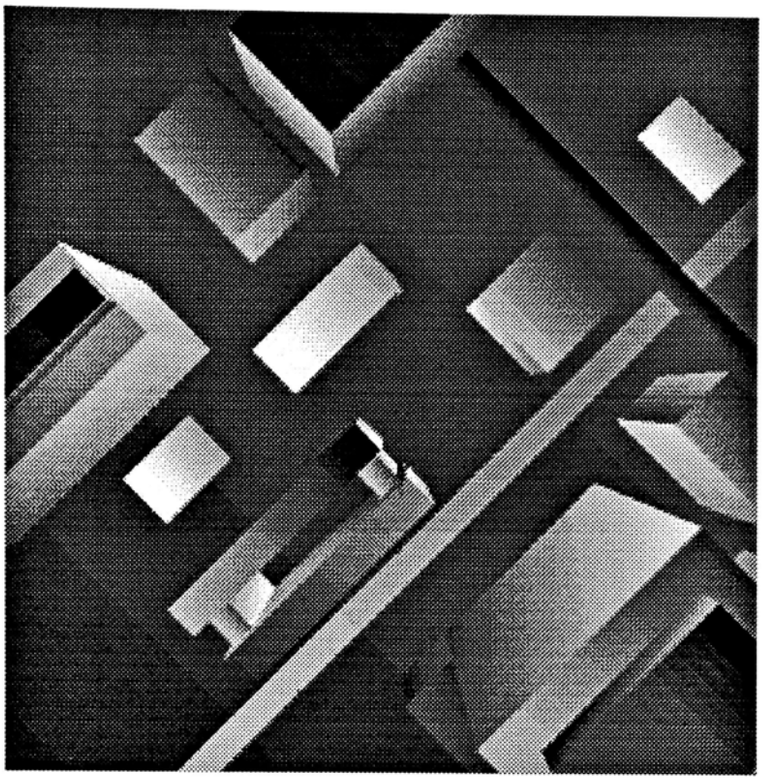


image 1

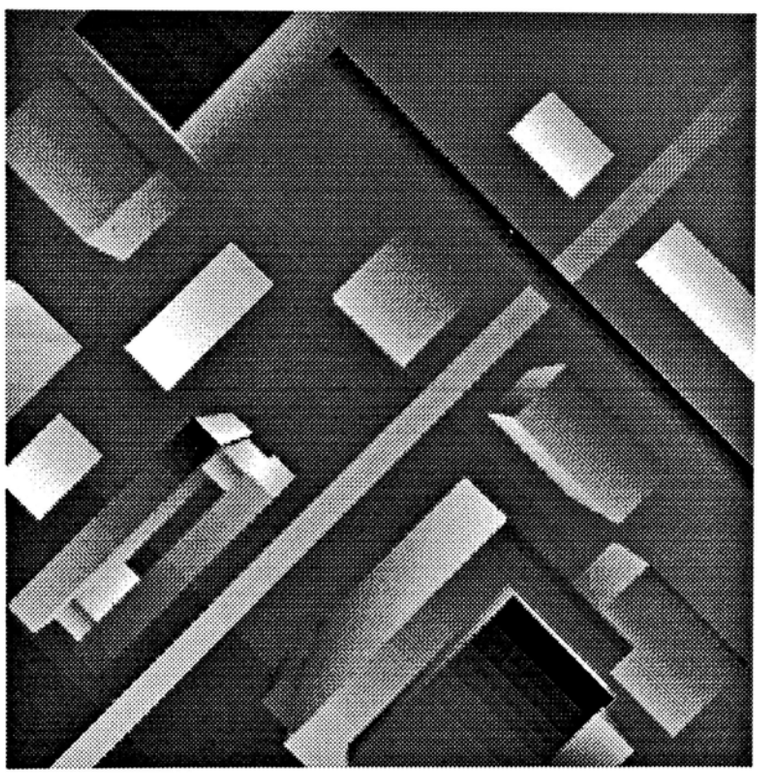
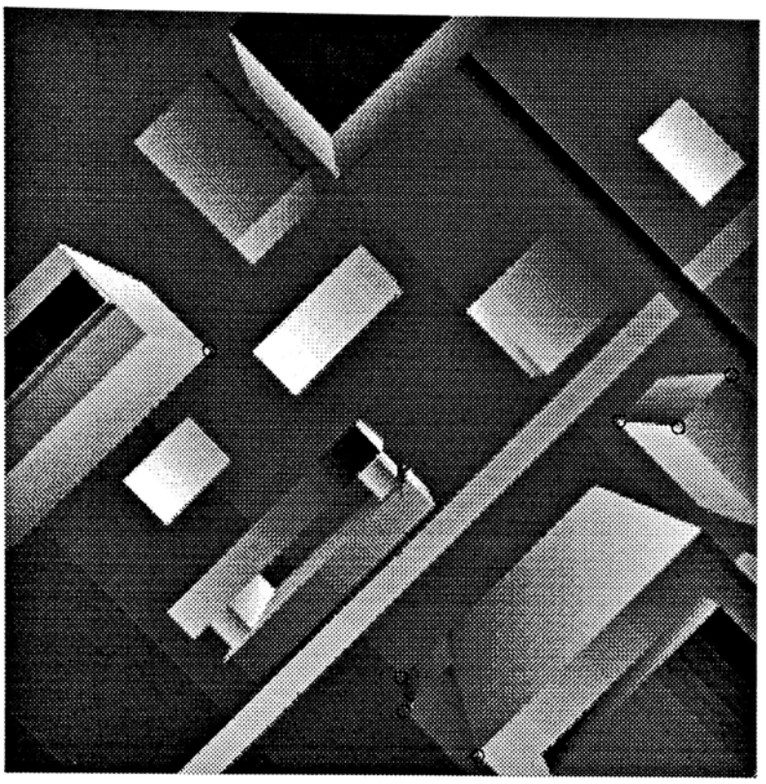
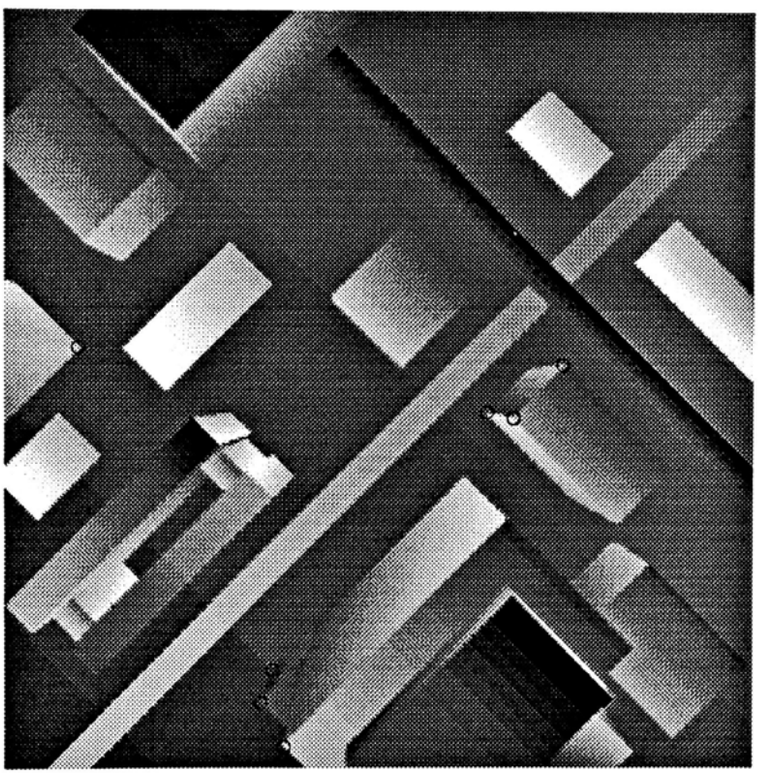


image 2

Figure 4.18: Stereo images of a synthetic scene.

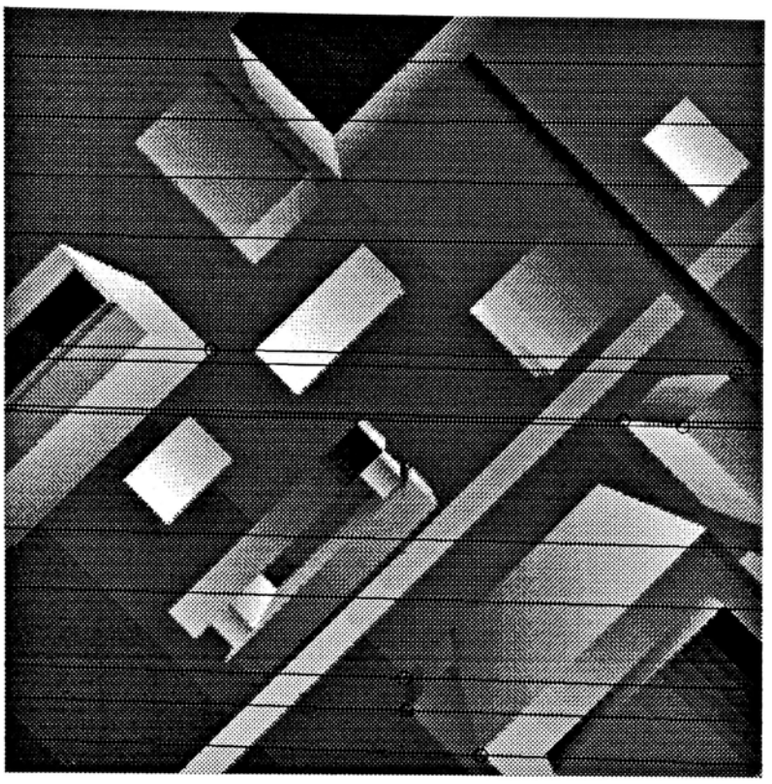


OTV correspondences used (image 1)

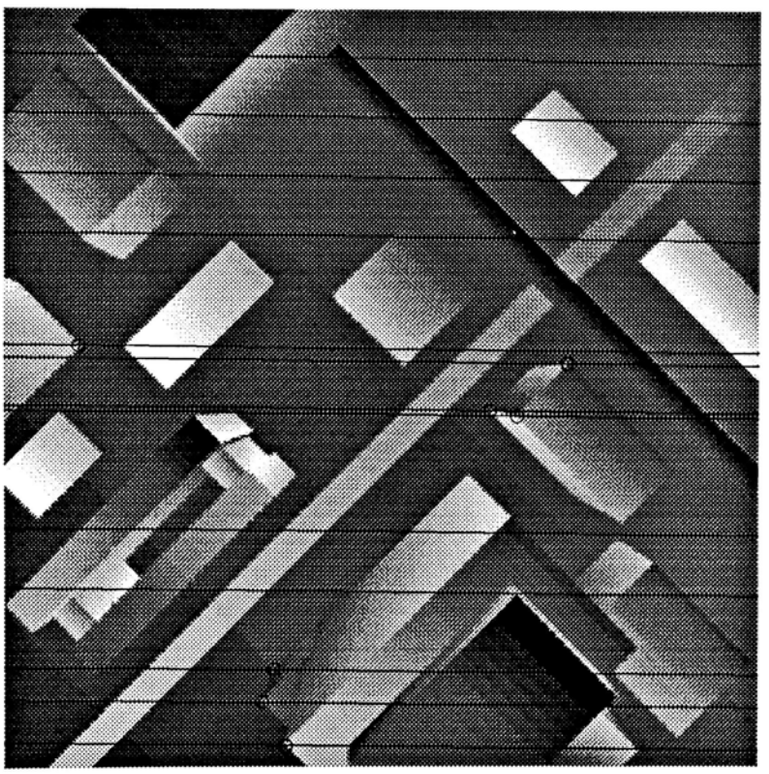


OTV correspondences used (image 2)

Figure 4.19: OTV correspondences used for the synthetic scene.



epipolar lines estimated (image 1)



epipolar lines estimated (image 2)

Figure 4.20: Estimated epipolar lines of the synthetic images.

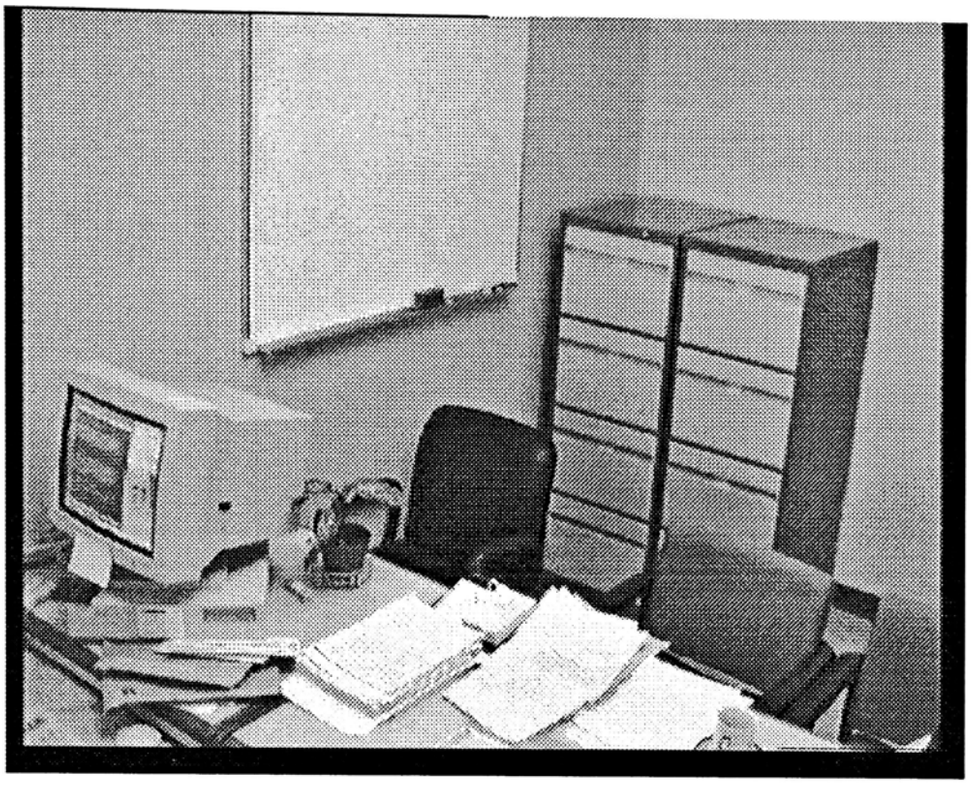


image 1

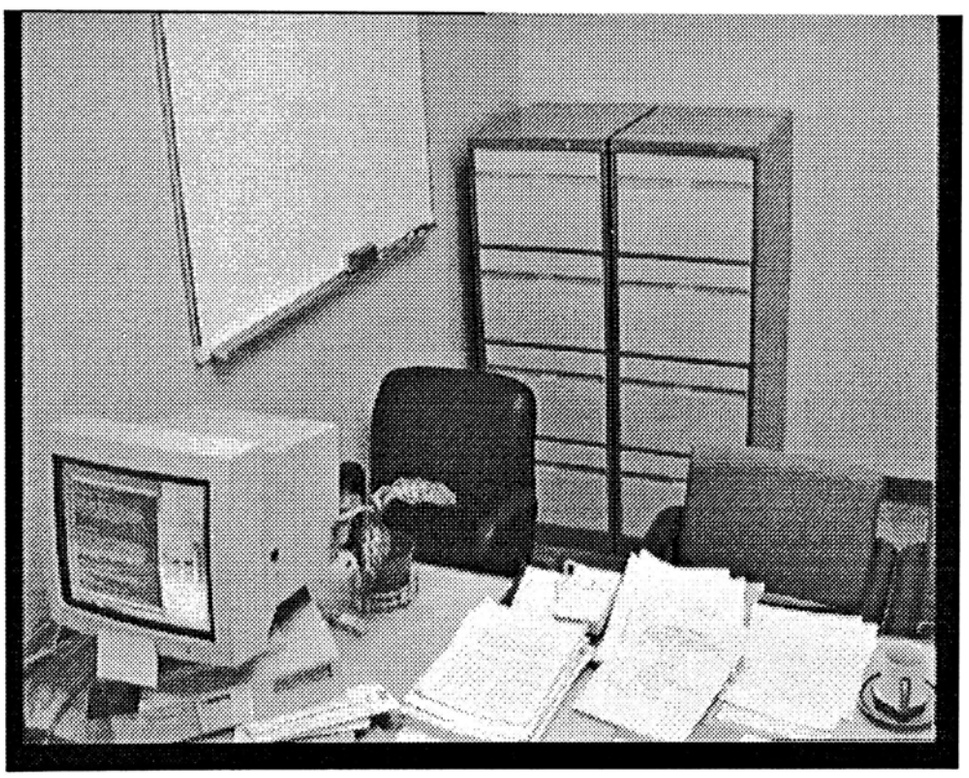
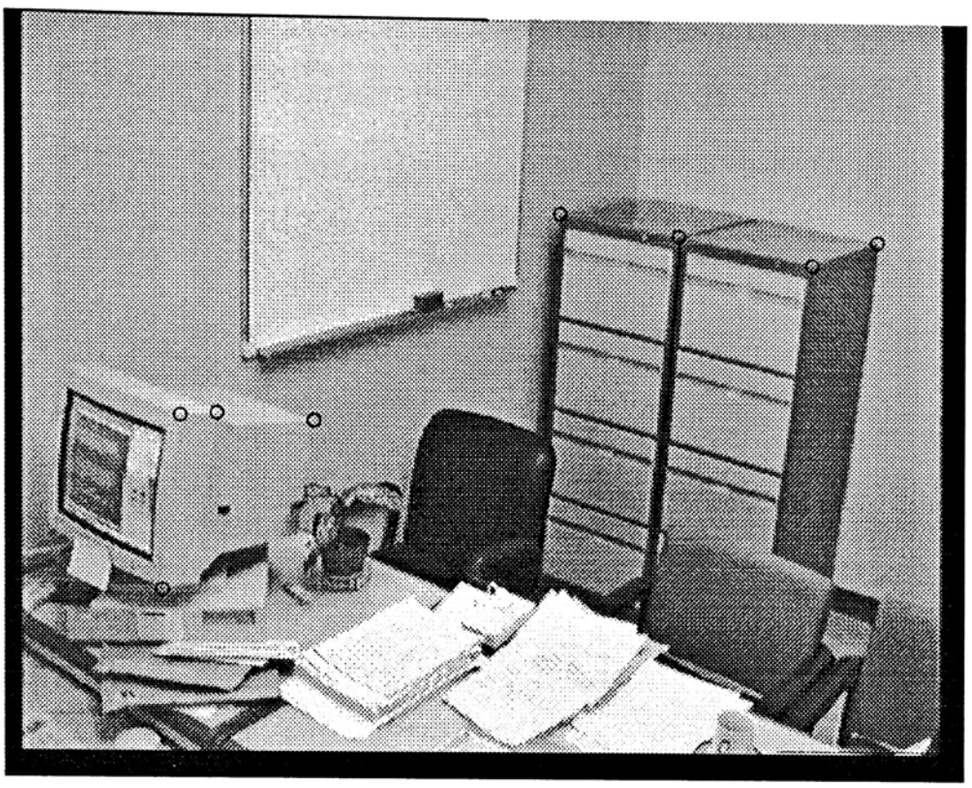
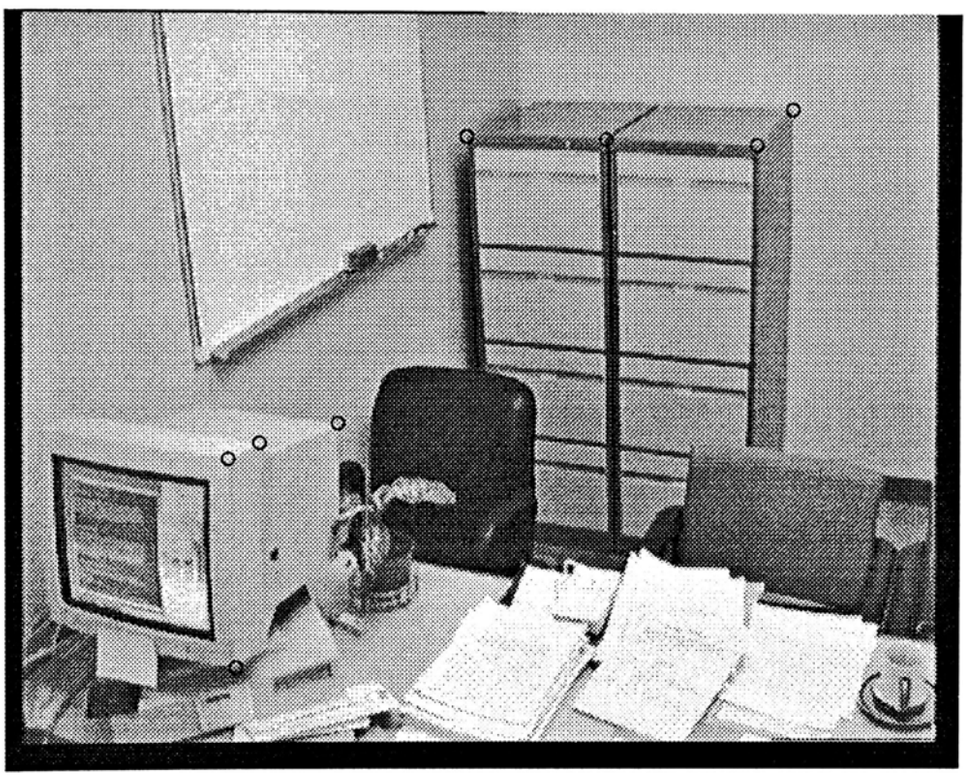


image 2

Figure 4.21: Stereo images of an indoor scene.

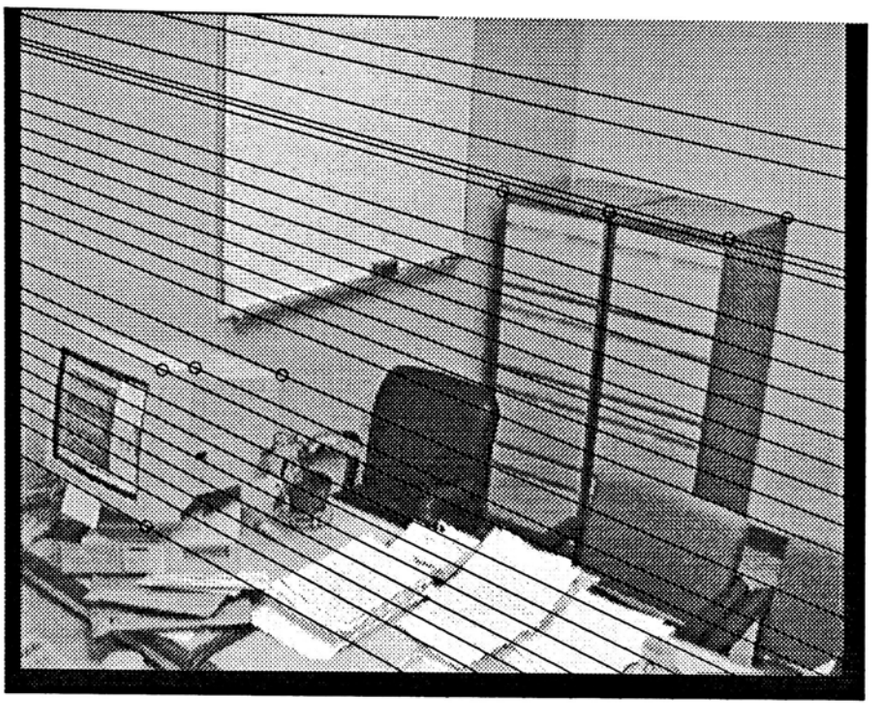


OTV correspondences used (image 1)

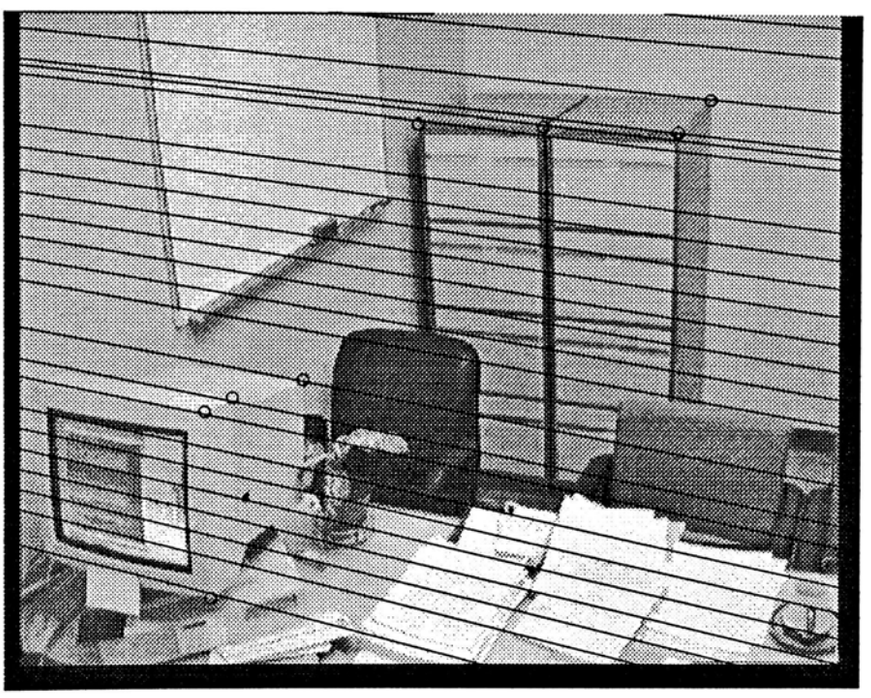


OTV correspondences used (image 2)

Figure 4.22: OTV correspondences used for the indoor scene.



epipolar lines estimated (image 1)



epipolar lines estimated (image 2)

Figure 4.23: Estimated epipolar lines of the indoor images.

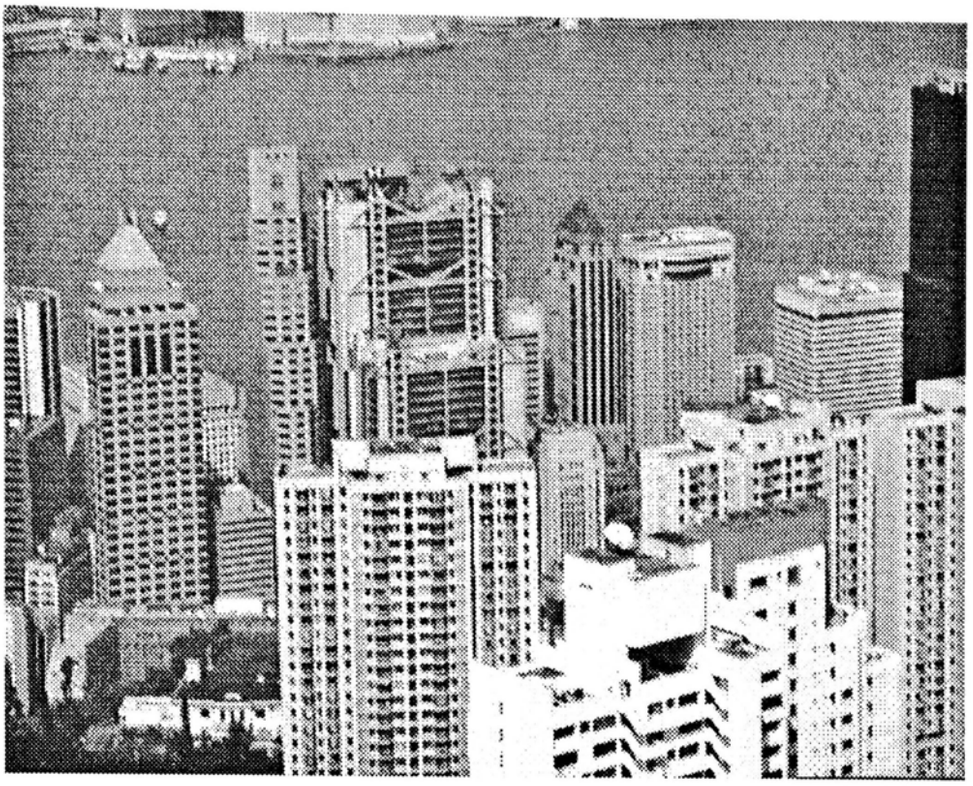


image 1

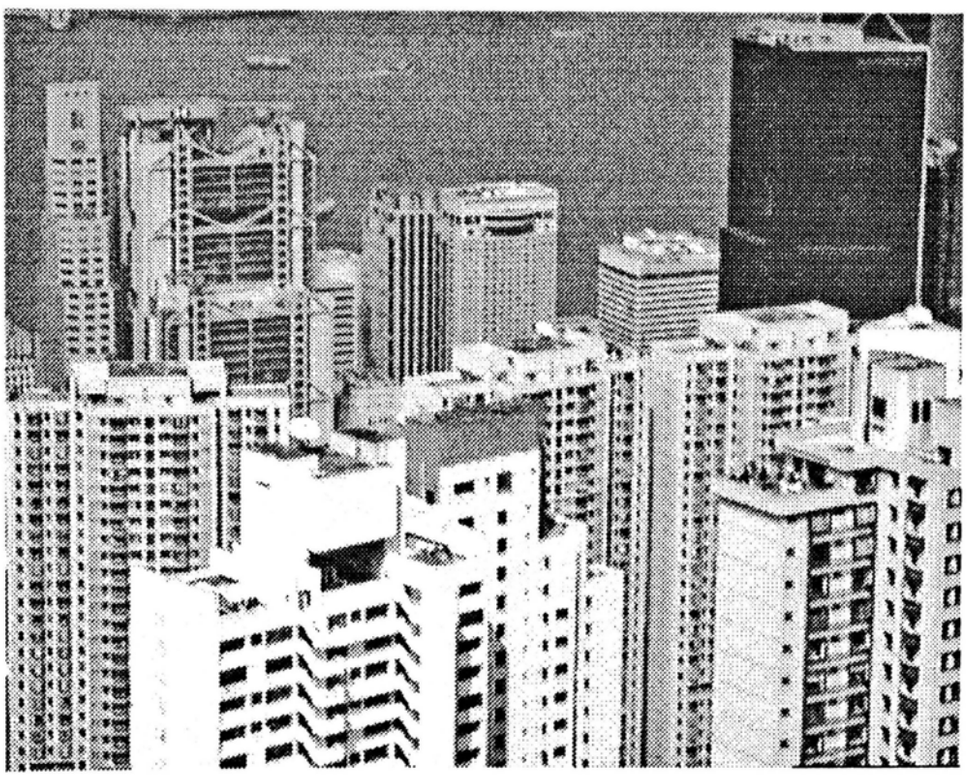
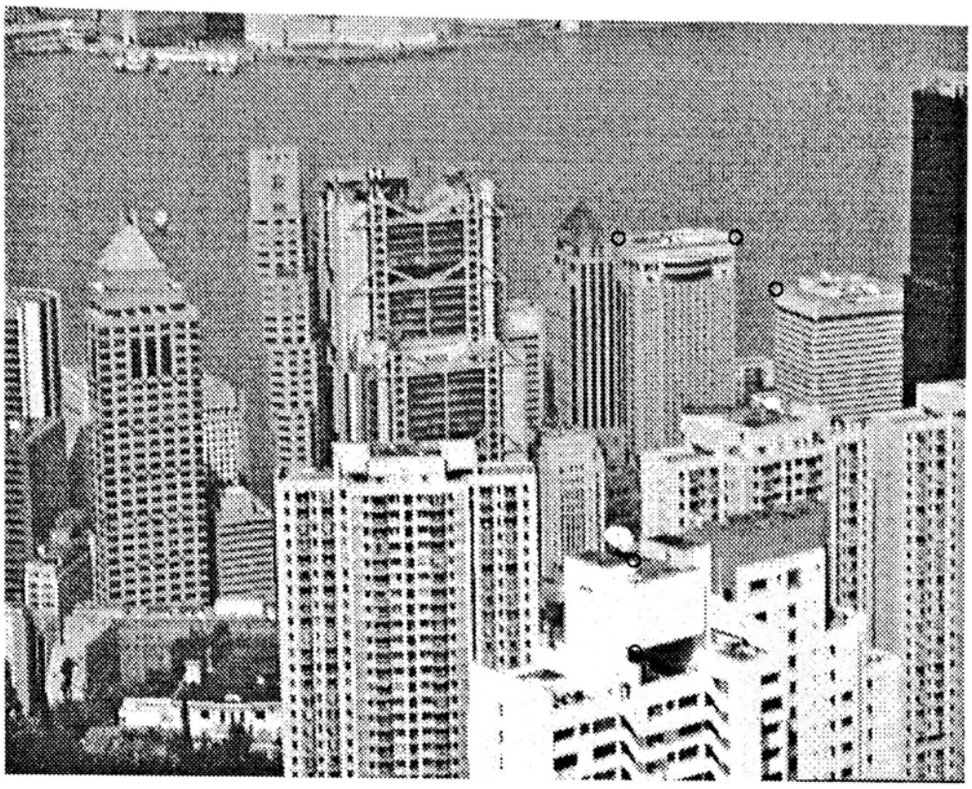
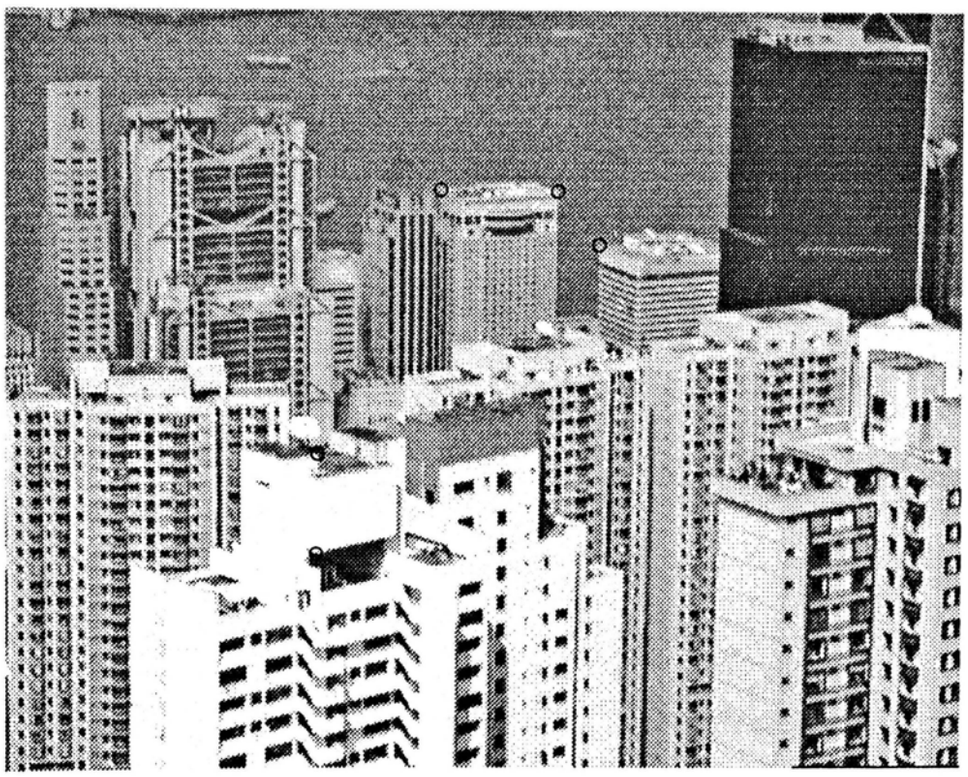


image 2

Figure 4.24: Stereo images of an aerial scene of Hong Kong.

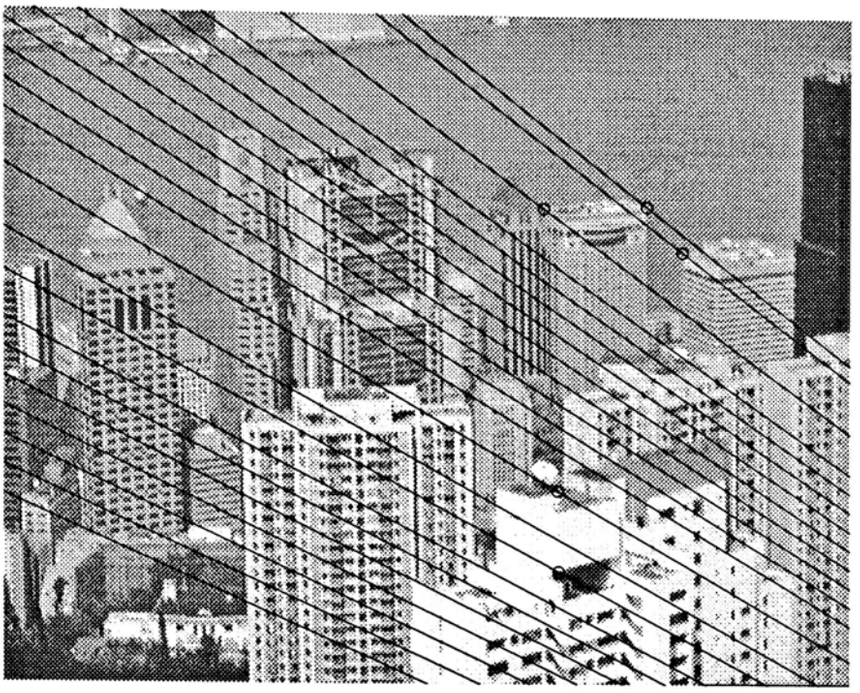


OTV correspondences used (image 1)

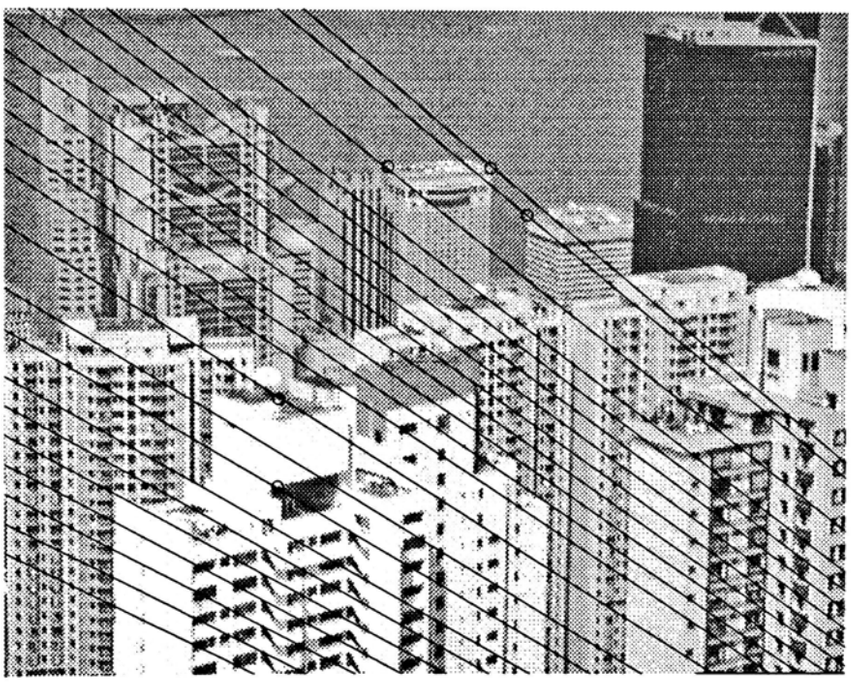


OTV correspondences used (image 2)

Figure 4.25: OTV correspondences used for the aerial scene.



epipolar lines estimated (image 1)



epipolar lines estimated (image 2)

Figure 4.26: Estimated epipolar lines of the aerial images.

Chapter 5

Error Analysis

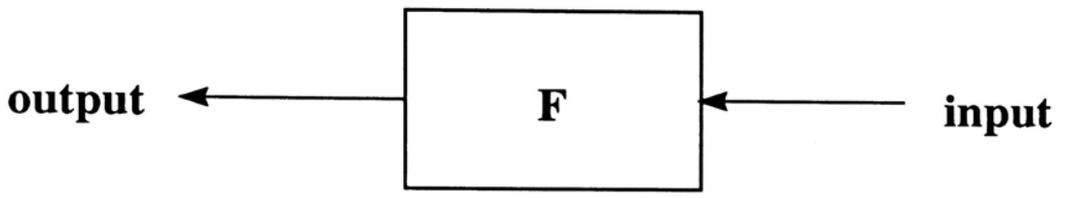


Figure 5.1: Calibration process modelled as an operator between I/O data.

The process of camera calibration can be viewed as a system with projected OTV features as input, and calculated extrinsic parameters as output. As shown in 5.1, for rotational parameters, *input* is a vector consists of one position component (projection of the vertex as u_{i0}, v_{i0}) plus three line vector components (expressed in θ_{ij}, where $i = 1, 2$ for 1st and 2nd image respectively; $j = 1, 2, 3$ for the corresponding 3 branches) of the two correspondingly projected OTV images, namely:

$$[u_{10} \quad v_{10} \quad \theta_{11} \quad \theta_{12} \quad \theta_{13} \quad u_{20} \quad v_{20} \quad \theta_{21} \quad \theta_{22} \quad \theta_{23}]^T$$

With extrinsic parameters output as:

$$[r_{11} \quad r_{12} \quad r_{13} \quad r_{21} \quad r_{22} \quad r_{23} \quad r_{31} \quad r_{32} \quad r_{33}]^T$$

where, r_{ij} is the i-th row j-th column element of the rotation matrix R. t_x, t_y, t_z are the components of the translation vector. Following from Section 3.3, suppose $\{\hat{L}_{i1}\}$ and $\{\hat{L}_{i2}\}$ are the directions of the OTV branches

computed from the two views separately, where $i = 1$, 2 or 3 is referring to the ith branch of the OTV.

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} L_{11x} & L_{21x} & L_{31x} \\ L_{11y} & L_{21y} & L_{31y} \\ L_{11z} & L_{21z} & L_{31z} \end{bmatrix} \begin{bmatrix} L_{12x} & L_{12y} & L_{12z} \\ L_{22x} & L_{22y} & L_{22z} \\ L_{32x} & L_{32y} & L_{32z} \end{bmatrix} \tag{5.1}$$

As has shown in Section 3.2, $\{\hat{L}_{ijk}\}$, with $k = x, y$ or z, can be expressed in the form of Equation 3.9. Equation 5.1 can be re-written in the form:

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \end{bmatrix} = \begin{bmatrix} F_1(u_{10}, v_{10}, \theta_{11}, \theta_{12}, \theta_{13}, u_{20}, v_{20}, \theta_{21}, \theta_{22}, \theta_{23}) \\ F_2(u_{10}, v_{10}, \theta_{11}, \theta_{12}, \theta_{13}, u_{20}, v_{20}, \theta_{21}, \theta_{22}, \theta_{23}) \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ F_9(u_{10}, v_{10}, \theta_{11}, \theta_{12}, \theta_{13}, u_{20}, v_{20}, \theta_{21}, \theta_{22}, \theta_{23}) \end{bmatrix}$$

F's are the function which maps the input to extrinsic parameters.

Under any particular configuration, the deviation of the output with respect to input can be related by:

$$\Delta\vec{o} = J(\vec{i})\Delta\vec{i}$$

where $J(\vec{i})$ is the Jacobian matrix of i with 9×10 entries.

$$J(\vec{i}) = \begin{bmatrix} \frac{\partial}{\partial u_{10}} F_1 & \frac{\partial}{\partial v_{10}} F_1 & \cdots & \frac{\partial}{\partial \theta_{23}} F_1 \\ \frac{\partial}{\partial u_{10}} F_2 & \frac{\partial}{\partial v_{10}} F_2 & \cdots & \frac{\partial}{\partial \theta_{23}} F_2 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial u_{10}} F_{12} & \frac{\partial}{\partial v_{10}} F_{12} & \cdots & \frac{\partial}{\partial \theta_{23}} F_{12} \end{bmatrix}$$

The translation parameters are calculated from the evaluated R and point correspondences. In real application, at least two OTVs are used for R recovery and the vertices are used as point correspondences for t recovery. In error analysis, using two OTVs complicates the task by introducing two candidates R for the functions which maps the inputs to t. Here, we represent the t_x, t_y, t_z as functions of R and two projected points with one of them comes from the

vertex $(u_{10}, v_{10}, u_{20}, v_{20})$ which was used to evaluate the R, and the other point $(u_{11}, v_{11}, u_{21}, v_{21})$.

$$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} F_{10}(r_{11}, r_{12}, r_{13}, r_{21}, \cdots, r_{32}, r_{33}, u_{10}, v_{10}, u_{20}, v_{20}, u_{11}, v_{11}, u_{21}, v_{21}) \\ F_{11}(r_{11}, r_{12}, r_{13}, r_{21}, \cdots, r_{32}, r_{33}, u_{10}, v_{10}, u_{20}, v_{20}, u_{11}, v_{11}, u_{21}, v_{21}) \\ F_{12}(r_{11}, r_{12}, r_{13}, r_{21}, \cdots, r_{32}, r_{33}, u_{10}, v_{10}, u_{20}, v_{20}, u_{11}, v_{11}, u_{21}, v_{21}) \end{bmatrix}$$

So the Jacobian matrix for translation is expressed as a 3 by 17 matrix:

$$J(\vec{t}) = \begin{bmatrix} \frac{\partial}{\partial r_{11}} F_{10} & \cdots & \frac{\partial}{\partial r_{33}} F_{10} & \frac{\partial}{\partial u_{10}} F_{10} & \cdots & \frac{\partial}{\partial v_{20}} F_{10} & \frac{\partial}{\partial u_{11}} F_{10} & \cdots & \frac{\partial}{\partial v_{21}} F_{10} \\ \frac{\partial}{\partial r_{11}} F_{11} & \cdots & \frac{\partial}{\partial r_{33}} F_{11} & \frac{\partial}{\partial u_{10}} F_{11} & \cdots & \frac{\partial}{\partial v_{20}} F_{11} & \frac{\partial}{\partial u_{11}} F_{11} & \cdots & \frac{\partial}{\partial v_{21}} F_{11} \\ \frac{\partial}{\partial r_{11}} F_{12} & \cdots & \frac{\partial}{\partial r_{33}} F_{12} & \frac{\partial}{\partial u_{10}} F_{12} & \cdots & \frac{\partial}{\partial v_{20}} F_{12} & \frac{\partial}{\partial u_{11}} F_{12} & \cdots & \frac{\partial}{\partial v_{21}} F_{12} \end{bmatrix}$$

Equations of $F_k, k = 1$ to 9, are lengthy. With the help of symbolic mathematics package[1], the task on derivation of the equation is greatly simplified. Unfortunately, the resulting equation , even with just one of the nine equations, shown as plain Mathematica format is so long that occupys 482 pages of A4 size paper. F_{10}, F_{11}, andF_{12} are also lengthy. It is impractical to present the resulting equations here. Instead, I had included examples of the resulting Jacobian matrix under three different configuration.

In the following sections, all configurations are based on the a camera model with $f = -1$ unit, image size is 256 pixels $\times$ 256 pixels in 2 units $\times$ 2 units. Quantization noise is then assumed to be $\pm 1/256$ unit. For the generation of a random OTV, refer to chapter 4 for the details.

In first case, we use translation in x direction without rotation. We choose this configuration because we have a synthetic image being tested in section 4.2 which also involves translation in x only. In the second and third cases, we use translation with rotation. We choose to use two set of R and $\hat{t}$ which had been used in chapter 4 for simulation.

Regarding to the meaning of the rotation and translation parameters, this chapter and the section 4.1 follow the convention from Weng *et al.* [33]. That is, rotation matrix and translational vector map the spatial vectors between two view. Therefore, you will see that a translation of $[6 \quad 0 \quad 0]^T$ with zero rotation looks a camera moving in $[-6 \quad 0 \quad 0]^T$ between the two views.

[1]Mathematica^{TM} is used here. Mathematica is a trademark of Wolfram Research Inc.

5.1 Translation in x only

In this configuration, no rotation is introduced, translation is being carried out as $[6 \quad 0 \quad 0]^T$. Figure 5.2 shows the projected OTV in two views. The OTV is randomly generated:

$$\begin{aligned} L_{11} &= (0.111643, \quad 0.993149, \quad -0.034501) \\ L_{21} &= (0.804990, \quad -0.070025, \quad 0.589140) \\ L_{31} &= (0.582688, \quad -0.093547, \quad -0.807294) \end{aligned}$$

with vertex at $(-2.375470, -4.525355, -13.3608188)$ according to the left camera coordinates. The 'o' is a randomly generated point correspondences for the recovery of t.

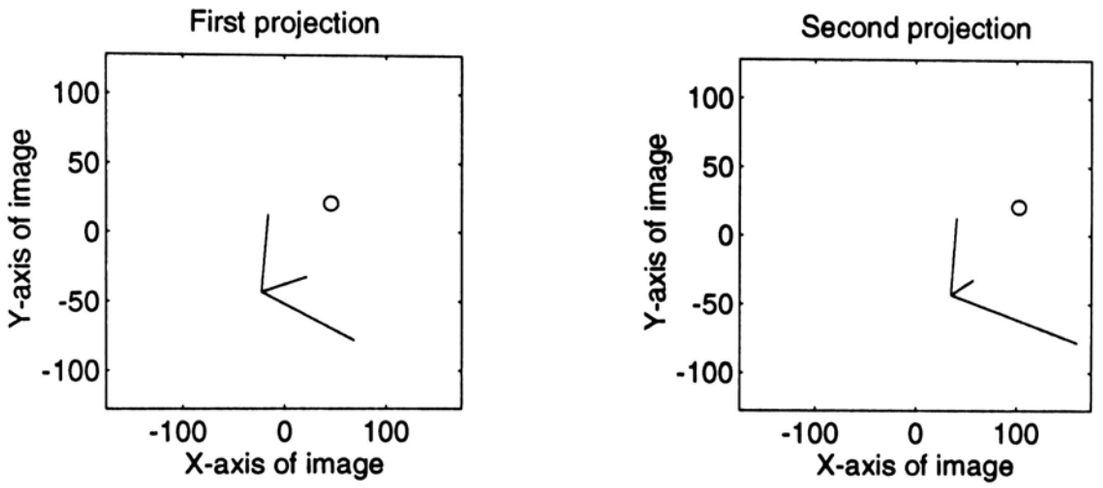


Figure 5.2: Projected images; no rotation, translation in x only.

5.1.1 Jacobian Matrix on Rotation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix}
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 \\
0.0208091 & 0.1381760 & -1.0422500 & -0.0447806 & 0.1045480 \\
-0.4177820 & 0.1490440 & -1.5652900 & 0.9326080 & 0.8006880 \\
-0.0208091 & -0.1381760 & 1.0422500 & 0.0447806 & -0.1045480 \\
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 \\
0.0395883 & -0.9932350 & 0.0016642 & 0.4451940 & -0.6368580 \\
0.4177820 & -0.1490440 & 1.5652900 & -0.9326080 & -0.8006880 \\
-0.0395883 & 0.9932350 & -0.0016642 & -0.4451940 & 0.6368580 \\
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000
\end{bmatrix}$$

$$\begin{bmatrix}
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 \\
-0.0203177 & 0.4047980 & 1.3437000 & -0.5092650 & 0.0626303 \\
0.4926390 & 0.6364990 & 1.9202800 & -1.9831400 & -0.2766650 \\
0.0203177 & -0.4047980 & -1.3437000 & 0.5092650 & -0.0626303 \\
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 \\
-0.0657046 & 1.1974800 & 0.6753880 & -1.2508600 & 0.2728710 \\
-0.4926390 & -0.6364990 & -1.9202800 & 1.9831400 & 0.2766650 \\
0.0657046 & -1.1974800 & -0.6753880 & 1.2508600 & -0.2728710 \\
0.0000000 & 0.0000000 & 0.0000000 & 0.0000000 & 0.0000000
\end{bmatrix}$$

Entries for mapping input variations to r_{11}, r_{22}, r_{33} are zero. It implies that, for zero rotation, the perturbation in input results little effect on the diagonal elements in the calculated rotation matrix. In other words, the configuration is at the turning points of F_1, F_5, F_9. We had done the simulation one more time with another randomly generated OTV, the rows of the Jacobian matrix corresponding to the diagonal elements are still all zero. Because the simulation process is lengthy[2], we had not tried it with even more OTVs.

Next, we substitute different Δ input parameter to see the effect in the output. Since quantization is up to $\pm\frac{1}{2}$ pixel, we try the $\Delta u_{10}, \Delta v_{10}, \Delta u_{20}, \Delta v_{20}$ with $\pm\frac{1}{2}$ pixel[3].

While the quantization error on point projection is obviously $\pm\frac{1}{2}$ pixel, it is not so straight forward for the deviation in the projected angle. The deviation depends on the length of projected branch and quantization error

[2]It takes 6 to 8 hours for one trial.

[3]$\frac{1}{2}$ pixel here means 1/256 unit length according to the camera model used.

on pixels along the edge. As an example, the first branch projected on left image in figure 5.2 is projected as an direction vector of (55.8707, 6.5486). With $+\frac{1}{2}$ pixel/$-\frac{1}{2}$ pixel along the x direction, the corresponding deviations in angle are $-0.0176rad/0.0177rad$. For simplicity, we use $\frac{1}{2}$ pixel deviation subtend on 60 pixels which corresponds to $\Delta\theta = 0.0083331rad$.

Table 5.1 shows the ΔR corresponding to variations in the input parameters. It is observed that, for rotation = 0, the deviation in output is in a form of skew symmetric matrix.

5.1.2 Jacobian Matrix on Translation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.41187 & -0.00318806 & 2.24618 & -0.1329 & -0.124613 \\ 0 & 0 & 0 & -2.38493 & -2.23622 & 0.0572105 & -0.996468 & 0.386313 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.00318806 & 0 & 0.728028 & 0 & -0.728028 & 0 & 1.51815 & 0 & -1.51815 \\ 2.23622 & 0 & -4.42504 & 0 & 4.42504 & 0 & 4.48225 & 0 & -4.48225 \end{bmatrix}$$

Table 5.2 shows the Δt corresponding to variations in the input parameters. It is found that the t_x component, the translation is taken along this direction only, is invariant with input perturbation when the perturbation is close to *zero*.

$\Delta input$	ΔR
$+\Delta u_{10}, +\Delta v_{10}$	$\begin{matrix} 0 & 0.00062 & -0.00105 \\ -0.00062 & 0 & -0.00373 \\ 0.00105 & 0.00373 & 0 \end{matrix}$
$-\Delta u_{10}, +\Delta v_{10}$	$\begin{matrix} 0 & 0.00046 & 0.00221 \\ -0.00046 & 0 & -0.00403 \\ -0.00221 & 0.00403 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	$\begin{matrix} 0 & 0.00212 & 0.00336 \\ -0.00212 & 0 & 0.00070 \\ -0.00336 & -0.00070 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	$\begin{matrix} 0 & -0.00088 & -0.00546 \\ 0.00088 & 0 & -0.00815 \\ 0.00546 & 0.00815 & 0 \end{matrix}$
$+\Delta\theta_{11}$	$\begin{matrix} 0 & -0.00869 & -0.01304 \\ 0.00869 & 0 & 0.00001 \\ 0.01304 & -0.00001 & 0 \end{matrix}$
$+\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}$	$\begin{matrix} 0 & -0.00819 & 0.00140 \\ 0.00819 & 0 & -0.00158 \\ -0.00140 & 0.00158 & 0 \end{matrix}$
$+\Delta\theta_{11}, +\Delta\theta_{12}, -\Delta\theta_{13}$	$\begin{matrix} 0 & -0.00993 & -0.01194 \\ 0.00993 & 0 & 0.00903 \\ 0.01194 & -0.00903 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}$	$\begin{matrix} 0 & -0.00757 & 0.00035 \\ 0.00757 & 0 & -0.00531 \\ -0.00035 & 0.00531 & 0 \end{matrix}$
$+\Delta u_{10}, -\Delta v_{10}, +\Delta\theta_{11}, +\Delta\theta_{12}, -\Delta\theta_{13}$	$\begin{matrix} 0 & -0.01039 & -0.01416 \\ 0.01039 & 0 & 0.01307 \\ 0.01416 & -0.01307 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, +\Delta\theta_{21}, +\Delta\theta_{22}, +\Delta\theta_{23}$	$\begin{matrix} 0 & 0.00141 & 0.00193 \\ -0.00141 & 0 & -0.00341 \\ -0.00193 & 0.00341 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, -\Delta\theta_{22}, -\Delta\theta_{23}$	$\begin{matrix} 0 & -0.01654 & -0.00123 \\ 0.01654 & 0 & -0.00721 \\ 0.00123 & 0.00721 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, -\Delta\theta_{22}, -\Delta\theta_{23}$	$\begin{matrix} 0 & -0.01354 & 0.00759 \\ 0.01354 & 0 & 0.00163 \\ -0.00759 & -0.00163 & 0 \end{matrix}$
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta\theta_{11}, -\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, +\Delta\theta_{22}, -\Delta\theta_{23}$	$\begin{matrix} 0 & -0.02444 & -0.04598 \\ 0.02444 & 0 & -0.03599 \\ 0.04598 & 0.03599 & 0 \end{matrix}$

Table 5.1: ΔR versus $\Delta input$.

$\Delta input$	Δt		
$+\Delta u_{10}, +\Delta v_{10}$	0.00000	-0.00525	-0.01562
$-\Delta u_{10}, +\Delta v_{10}$	0.00000	-0.00578	-0.01266
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	0.00000	0.00509	0.00818
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	0.00000	-0.01559	-0.03943
$+\Delta \theta_{11}$	0.00000	-0.01396	-0.03372
$+\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	0.00000	-0.01513	-0.01761
$+\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	0.00000	0.00580	-0.03856
$+\Delta u_{10}, +\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	0.00000	-0.02038	-0.03323
$+\Delta u_{10}, -\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	0.00000	0.01158	-0.02590
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, +\Delta \theta_{21}, +\Delta \theta_{22}, +\Delta \theta_{23}$	0.00000	-0.00583	0.00641
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	0.00000	-0.03492	-0.07288
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	0.00000	-0.01423	-0.02527
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, -\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, +\Delta \theta_{22}, -\Delta \theta_{23}$	0.00000	-0.12025	-0.12684

Table 5.2: Δt versus $\Delta input$.

5.2 Rotation + translation in x, y, z

In this configuration, rotation is taken about an axis (0.9, 1, -0.8) with rotation angle 5°; translation is (-0.5, 0.5, 3). Figure 5.3 shows the projected OTV in two views. The OTV used is same as in section 5.1.

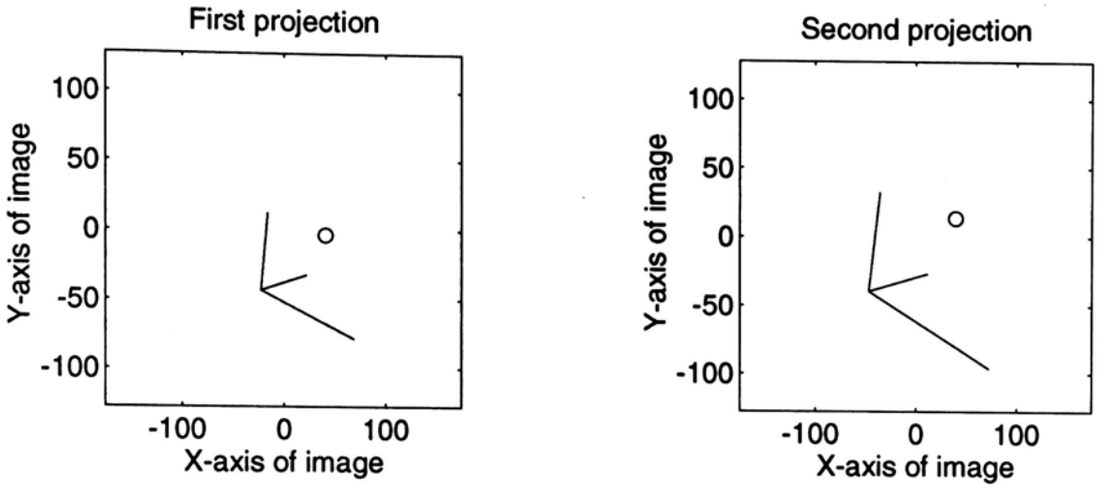


Figure 5.3: Projected images; with rotation, translation in x, y, z .

5.2.1 Jacobian Matrix on Rotation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix} -0.0218396 & 0.0144806 & -0.1332920 & 0.0488290 & 0.0484916 \\ 0.0422179 & 0.1302100 & -0.9595140 & -0.0925749 & 0.0631922 \\ -0.4155900 & 0.1553780 & -1.6118300 & 0.9277980 & 0.8035470 \\ -0.0185960 & -0.1920180 & 1.0396800 & 0.0689578 & -0.1390310 \\ -0.0011352 & 0.0569707 & -0.0450559 & -0.0247956 & 0.0372175 \\ 0.0406589 & -0.9825950 & -0.0575402 & 0.4413990 & -0.6291290 \\ 0.4148990 & -0.1030320 & 1.5612300 & -0.9506870 & -0.7693890 \\ -0.0575254 & 0.9974290 & -0.0691991 & -0.4039510 & 0.6699710 \\ -0.0256647 & 0.0570061 & -0.0889901 & 0.0312151 & 0.0766030 \end{bmatrix}$$

$$\begin{bmatrix} 0.0247709 & -0.0324873 & 0.1176500 & -0.0349494 & -0.0591101 \\ -0.0015645 & -0.2432980 & 0.9401910 & 0.1288570 & -0.1600740 \\ 0.4361850 & -0.3856830 & 1.3518200 & -0.7116240 & -0.9164200 \\ -0.0231158 & 0.3191160 & -1.0114000 & -0.1131420 & 0.2506480 \\ 0.0030430 & -0.0647146 & 0.0477283 & 0.0239250 & -0.0503168 \\ -0.0403937 & 1.0212100 & -0.0236081 & -0.3820870 & 0.7916320 \\ -0.4333630 & 0.3251860 & -1.2990900 & 0.7341640 & 0.8695180 \\ 0.0604330 & -1.0528700 & 0.1373530 & 0.3565540 & -0.8430670 \\ 0.0268248 & -0.0720169 & 0.0781567 & -0.0218086 & -0.0909964 \end{bmatrix}$$

Table 5.3 shows the ΔR corresponding to variations in the input parameters.

5.2.2 Jacobian Matrix on Translation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix} -0.820446 & -0.390806 & 1.43337 & 0.949553 & 0.668321 \\ -0.647431 & -0.441312 & -0.473176 & 1.49988 & 1.2232 \\ -0.0556886 & 0.00163118 & 0.437143 & -0.094548 & -0.102039 \end{bmatrix}$$

$$\begin{bmatrix} 0.948307 & -0.393994 & -0.233434 & 0.136016 & -0.727747 \\ 3.52018 & -0.469908 & -0.362851 & -0.856933 & -1.51323 \\ -0.522822 & 0.00693982 & 0.0219994 & 0.213503 & 0.149512 \end{bmatrix}$$

$$\begin{bmatrix} 2.36823 & 0.509376 & -1.89233 & 3.03793 & -1.36195 \\ 4.92436 & 1.05917 & -3.9348 & 1.07188 & -0.480539 \\ -0.48654 & -0.104649 & 0.388769 & 0.486978 & -0.21832 \end{bmatrix}$$

$$\begin{bmatrix} -1.99009 & 1.01499 \\ -0.702166 & 0.35812 \\ -0.31901 & 0.162702 \end{bmatrix}$$

Table 5.4 shows the Δt corresponding to variations in the input parameters.

$\Delta input$	ΔR		
$+\Delta u_{10}, +\Delta v_{10}$	-0.00003	0.00067	-0.00102
	-0.00082	0.00022	-0.00368
	0.00122	0.00367	0.00012
$-\Delta u_{10}, +\Delta v_{10}$	0.00014	0.00034	0.00223
	-0.00068	0.00023	-0.00400
	-0.00202	0.00412	0.00032
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	-0.00006	-0.00028	-0.00082
	0.00033	-0.00002	0.00015
	0.00080	-0.00021	-0.00005
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	0	0.00163	-0.00121
	-0.00198	0.00046	-0.00751
	0.00164	0.00755	0.00030
$+\Delta \theta_{11}$	-0.00111	-0.00800	-0.01343
	0.00866	-0.00038	-0.00048
	0.01301	-0.00058	-0.00074
$+\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	-0.00030	-0.00824	0.00100
	0.00808	-0.00027	-0.00204
	-0.00132	0.00164	0.00016
$+\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	-0.00111	-0.00929	-0.01240
	0.01040	-0.00089	0.00844
	0.01150	-0.00953	-0.00112
$+\Delta u_{10}, +\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	-0.00033	-0.00757	-0.00002
	0.00726	-0.00005	-0.00572
	-0.00011	0.00531	0.00028
$+\Delta u_{10}, -\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	-0.00125	-0.00964	-0.01463
	0.01107	-0.00112	0.01244
	0.01352	-0.01365	-0.00144
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, +\Delta \theta_{21}, +\Delta \theta_{22}, +\Delta \theta_{23}$	-0.00016	-0.00095	-0.00213
	0.00113	-0.00012	0.00132
	0.00201	-0.00147	-0.00019
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	-0.00049	-0.01419	0.00208
	0.01338	0.00001	-0.01277
	-0.00222	0.01210	0.00074
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	-0.00056	-0.01610	0.00248
	0.01570	-0.00047	-0.00511
	-0.00307	0.00434	0.00039
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, -\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, +\Delta \theta_{22}, -\Delta \theta_{23}$	-0.00170	-0.01051	-0.02183
	0.01017	0.00085	-0.02681
	0.02247	0.02524	0.00007

Table 5.3: ΔR versus $\Delta input$.

$\Delta input$	Δt		
$+\Delta u_{10}, +\Delta v_{10}$	-0.00073	-0.00240	0.00034
$-\Delta u_{10}, +\Delta v_{10}$	0.00495	-0.00360	0.00193
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	0.00016	0.00340	-0.00068
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	-0.00163	-0.00820	0.00135
$+\Delta \theta_{11}$	-0.01279	0.01618	-0.00643
$+\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	0.01061	0.00784	0.00083
$+\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	0.00159	0.05391	-0.01105
$+\Delta u_{10}, +\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	0.00987	0.00544	0.00117
$+\Delta u_{10}, -\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	-0.00904	0.04570	-0.01181
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, +\Delta \theta_{21}, +\Delta \theta_{22}, +\Delta \theta_{23}$	0.00024	0.00959	-0.00197
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	0.01950	0.00130	0.00431
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	0.02130	0.01289	0.00227
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, -\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, +\Delta \theta_{22}, -\Delta \theta_{23}$	-0.03995	-0.04898	0.00099

Table 5.4: Δt versus $\Delta input$.

5.3 Rotation + translation in x, y

In this configuration, rotation is taken about an axis (1, -0.2, -0.2) with rotation angle 8°; translation is (1, 3, 0). Figure 5.4 shows the projected OTV in two views. The OTV is randomly generated:

$$\begin{aligned} L_{11} &= (0.4415171, \quad 0.8513583, \quad 0.2832872) \\ L_{21} &= (-0.3393345, \quad -0.1338397, \quad 0.9310956) \\ L_{31} &= (0.8306111, \quad -0.5072238, \quad 0.2298027) \end{aligned}$$

with vertex at $(-2.8104081, -4.5295538, -12.7886472)$ according to the left camera coordinates.

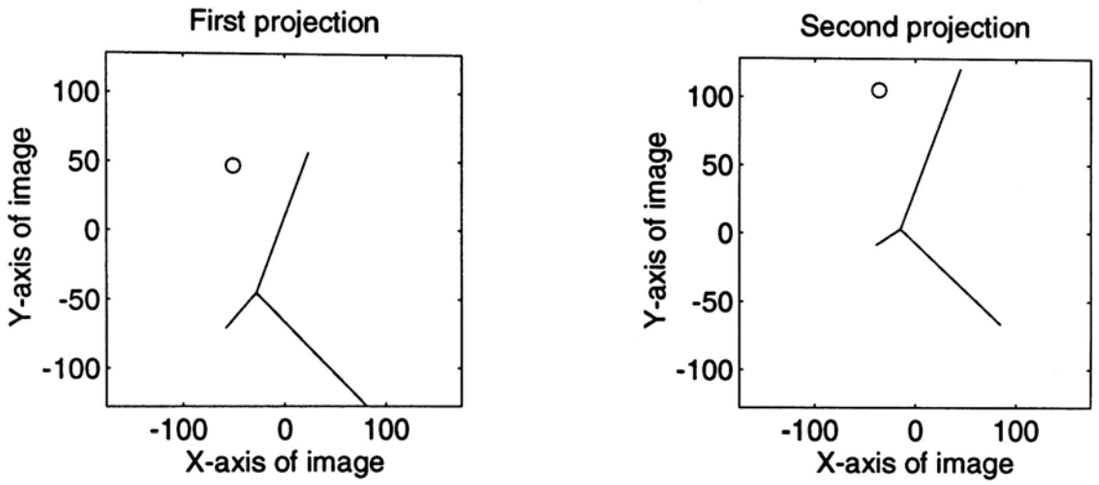


Figure 5.4: Projected images; with rotation, translation in x, y .

5.3.1 Jacobian Matrix on Rotation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix}
0.0102716 & 0.0103977 & 0.0211280 & 0.0039818 & -0.0514445 \\
-0.0303841 & 0.1189460 & -0.4341950 & -0.0377185 & -0.4818140 \\
-0.4456360 & -0.2798050 & -1.3419700 & -0.2024330 & 1.5064700 \\
0.0750817 & -0.0421781 & 0.6346800 & 0.0982547 & 0.2304830 \\
-0.0704803 & 0.1811670 & 0.1008780 & 0.1563240 & -0.2210120 \\
0.5059890 & -1.3286800 & -0.8622880 & -1.1715400 & 1.5876100 \\
0.4246770 & 0.3266110 & 1.2932200 & 0.2250550 & -1.5969200 \\
-0.5162610 & 1.3182800 & 0.8411600 & 1.1675600 & -1.5361600 \\
-0.0573488 & 0.1871590 & 0.1507250 & 0.1639070 & -0.2531900
\end{bmatrix}$$

$$\begin{bmatrix}
-0.0209112 & 0.0015217 & -0.0319828 & 0.0024966 & 0.0577777 \\
0.0068700 & -0.0740140 & 0.2896100 & 0.0547212 & 0.6464750 \\
0.8443210 & -0.1455600 & 1.6107200 & -0.0372476 & -1.5713900 \\
-0.1173190 & 0.0656495 & -0.5192980 & -0.0661276 & -0.4001980 \\
0.0139494 & -0.1293240 & -0.0663544 & -0.0775229 & 0.1540980 \\
-0.0810837 & 0.9418540 & 0.5861300 & 0.5842160 & -1.0623800 \\
-0.8335170 & 0.1105640 & -1.5718400 & 0.0294827 & 1.6699900 \\
0.1019950 & -0.9433760 & -0.5541480 & -0.5867120 & 1.0046000 \\
-0.0103012 & -0.1239980 & -0.1200690 & -0.0782512 & 0.1836330
\end{bmatrix}$$

Table 5.5 shows the ΔR corresponding to variations in the input parameters.

5.3.2 Jacobian Matrix on Translation

The Jacobian matrix is evaluated as:

$$\begin{bmatrix}
0.308798 & 0.745479 & 4.15312 & -0.0804142 & -0.291922 \\
-0.238187 & 0.632216 & -0.948388 & 0.0896427 & -0.241159 \\
-1.00573 & 6.00024 & 2.21722 & 0.454703 & -2.30488
\end{bmatrix}$$

$$\begin{bmatrix}
-1.26419 & 0.0957508 & -0.45157 & 0.012477 & 4.98855 \\
0.350913 & 0.118945 & -0.338202 & 0.431594 & -0.174144 \\
-0.240469 & 1.03417 & -3.32213 & 3.03969 & 9.40045
\end{bmatrix}$$

$$\begin{bmatrix}
-1.59666 & -5.1769 & 1.72563 & -1.10716 & 0.42277 \\
0.0557373 & 0.180719 & -0.0602395 & -0.805823 & 0.307704 \\
-3.00875 & -9.75536 & 3.25179 & -7.98196 & 3.04791
\end{bmatrix}$$

$\Delta input$	ΔR		
$+\Delta u_{10}, +\Delta v_{10}$	0.00008	0.00035	−0.00283
	0.00013	0.00043	−0.00321
	0.00293	0.00313	0.00051
$-\Delta u_{10}, +\Delta v_{10}$	0	0.00058	0.00065
	−0.00046	0.00098	−0.00717
	−0.00038	0.00717	0.00096
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	0	0.00008	−0.00010
	−0.00007	−0.00002	0.00015
	0.00011	−0.00015	−0.00002
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	0.00016	0.00061	−0.00556
	0.00033	0.00088	−0.00658
	0.00576	0.00642	0.00103
$+\Delta\theta_{11}$	0.00018	−0.00362	−0.01118
	0.00529	0.00084	−0.00719
	0.01078	0.00701	0.00126
$+\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}$	−0.00022	−0.00795	−0.00032
	0.00803	0.00030	−0.00372
	−0.00066	0.00394	0.00051
$+\Delta\theta_{11}, +\Delta\theta_{12}, -\Delta\theta_{13}$	0.00064	0.00008	−0.02542
	0.00419	0.00399	−0.03018
	0.02596	0.02954	0.00473
$+\Delta u_{10}, +\Delta v_{10}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}$	−0.00014	−0.00760	−0.00315
	0.00816	0.00073	−0.00693
	0.00228	0.00707	0.00102
$+\Delta u_{10}, -\Delta v_{10}, +\Delta\theta_{11}, +\Delta\theta_{12}, -\Delta\theta_{13}$	0.00064	−0.00050	−0.02607
	0.00465	0.00300	−0.02301
	0.02634	0.02237	0.00378
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, +\Delta\theta_{21}, +\Delta\theta_{22}, +\Delta\theta_{23}$	0.00002	0.00039	−0.00040
	−0.00026	0.00037	−0.00267
	0.00052	0.00265	0.00037
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, -\Delta\theta_{22}, -\Delta\theta_{23}$	−0.00030	−0.01560	−0.00590
	0.01657	0.00110	−0.01119
	0.00404	0.01149	0.00167
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta\theta_{11}, +\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, -\Delta\theta_{22}, -\Delta\theta_{23}$	−0.00045	−0.01612	−0.00044
	0.01617	0.00020	−0.00447
	−0.00161	0.00492	0.00062
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta\theta_{11}, -\Delta\theta_{12}, +\Delta\theta_{13}, -\Delta\theta_{21}, +\Delta\theta_{22}, -\Delta\theta_{23}$	−0.00049	−0.01400	0.00345
	0.01292	−0.00269	0.01743
	−0.00573	−0.01695	−0.00245

Table 5.5: ΔR versus $\Delta input$.

$$\begin{bmatrix} 1.05871 & -0.352902 \\ 0.770557 & -0.256852 \\ 7.63264 & -2.54421 \end{bmatrix}$$

Table 5.6 shows the Δt corresponding to variations in the input parameters.

$\Delta input$	Δt		
$+\Delta u_{10}, +\Delta v_{10}$	0.00456	0.00071	0.01468
$-\Delta u_{10}, +\Delta v_{10}$	0.02192	-0.00556	0.00785
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}$	-0.00070	0.00027	0.00037
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}$	0.00982	0.00116	0.02899
$+\Delta \theta_{11}$	-0.04279	0.00548	-0.05281
$+\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	-0.00517	-0.00652	-0.05652
$+\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	-0.07947	0.00798	-0.11345
$+\Delta u_{10}, +\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}$	-0.00061	-0.00581	-0.04184
$+\Delta u_{10}, -\Delta v_{10}, +\Delta \theta_{11}, +\Delta \theta_{12}, -\Delta \theta_{13}$	-0.06242	0.01217	-0.04786
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, +\Delta \theta_{21}, +\Delta \theta_{22}, +\Delta \theta_{23}$	0.00054	-0.00109	-0.00645
$+\Delta u_{10}, +\Delta v_{10}, -\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	-0.00176	-0.01052	-0.07723
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, +\Delta v_{20}, +\Delta \theta_{11}, +\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, -\Delta \theta_{22}, -\Delta \theta_{23}$	-0.01228	-0.01142	-0.10584
$+\Delta u_{10}, +\Delta v_{10}, +\Delta u_{20}, -\Delta v_{20}, +\Delta \theta_{11}, -\Delta \theta_{12}, +\Delta \theta_{13}, -\Delta \theta_{21}, +\Delta \theta_{22}, -\Delta \theta_{23}$	-0.02518	0.00038	-0.05091

Table 5.6: Δt versus $\Delta input$.

Chapter 6

Conclusion and Future work

Stereo calibration is a fundamental problem that has to be solved for any stereo vision system to be useful. However, the problem is non-trivial. The proposed method is particularly attractive as it does not require a rigid stereo camera setup to be established before pictures are taken. Stereo pictures can be taken at arbitrary viewpoints first, and calibration can be done later, as long as there are OTVs, some common features inside and outside man-made structures, visible in the scene. Aerial image understanding [12] via stereo vision is an immediate application of the work. The proposed method can also be applied to solve the absolute orientation problem of a rigid stereo camera setup, with aid of a reference object having rectangular structure components. The calibration solution can be estimated merely from the orthogonal properties of the corners, even if the exact dimensions of the reference object are unknown.

We have described how the orthogonality constraints, the vector projection constraint, the A-junction opacity constraint, and the orientation consistency constraint can be used together to compute R, and how epipolar constraints can be used to determine $\hat{t}$ between the stereo cameras. As opposed to most other methods that estimate the essential matrix E first ($E = \hat{t}_X R$ where

$$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}_X = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

) and then R and $\hat{t}$ iteratively, the proposed method is a closed-form solution that estimates R and $\hat{t}$ directly. It also requires only 2 OTV correspondences to recover uniquely all the parameters that are recoverable in the problem.

Experiments show that the calibration solution acquired from OTV correspondences is more stable than those acquired from point correspondences,

even with a few number of OTV correspondences. Experiments also show that unlike other methods, the accuracy in estimating R and $\hat{t}$ using the proposed method is generally insensitive to the direction and the magnitude of the translation and of the rotation.

We have used eye-picked coordinates in all the experiments. In fact, output from edge detection and line extraction algorithms can be employed to further increase the accuracy, while n-point approach cannot benefit from these image processing tools.

Future work can be an extension of the idea to use projections of OTVs with only two branches visible. The idea is also not restricted to be using orthogonal vertices only, and can be further generalized to use generic trihedral vertices, provided that their angular relationships are known *a priori* .

The angle constraint is not exclusive, in the sense that it can be used with the position constraint to compute an accurate R and $\hat{t}$. The simplest way is to compute R using OTV correspondences and $\hat{t}$ using all the available point correspondences.

Appendix A

Least-squares Approximation of a set of Rotation Matrices

Here we give a brief description of how we find the rotation matrix closest to a given set of rotation matrices $\{S_i\}$. What is difficult is to make sure whatever solution we have, it has to satisfy the orthonormal property of a rotation matrix.

We first transform each rotation matrix S_i to an equivalent unit quaternion representation $\hat{q}_i$ (please refer to [13, 5] for the definition and operations of quaternions, and their relationships with rotations). We work in the quaternion space instead of in the 3×3 matrix space, as it is easier to enforce the unity magnitude property of a quaternion than to enforce the orthonormal property of a 3×3 matrix, so that a true rotation representation in the final solution is assured. We use the following relation between rotation matrix R and unit quaternion $\hat{q} = [q_0 \ q_1 \ q_2 \ q_3]^T$ to transform one to the other:

$$R = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_1 + q_0 q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_3 q_1 - q_0 q_2) & 2(q_3 q_2 + q_0 q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix}$$

We use a least-squares method to locate the resultant rotation: find $\bar{q}$ of unity magnitude such that the sum

$$\sum_i (\hat{q}_i \cdot \bar{q})^2$$

is maximum. Notice that $\hat{q}$ and $-\hat{q}$ represent the same rotation. However, using the above formulation the *signs* of $\hat{q}_i$'s would not matter.

The problem can in fact be converted into a standard form: find $\bar{q}$ of unity magnitude that maximizes $\|A\bar{q}\|$, where

$$A = \begin{bmatrix} \hat{q}_1^T \\ \hat{q}_2^T \\ \cdots \\ \hat{q}_n^T \end{bmatrix}$$

The optimal solution for $\bar{q}$ is the unit eigenvector of the matrix A^TA associated with its largest eigenvalue. The rotation matrix transformed from the optimal $\bar{q}$ is the least-squares approximation we need, and the largest eigenvalue of A^TA is a measure of how *close* the given S_i's are to one another.

Appendix B

Epipolar Lines independent of the Translation Magnitude

Here we show how we compute epipolar lines over a stereo pair of images. The method also proves that epipolar lines are independent of the translation magnitude.

Let F_1 and F_2 be the focal points of the camera coordinate frames, and f_1 and f_2 be the respective focal lengths. Also let ${}^1R_2$ and $k\hat{t}$ be the rotational and translational relationship between the camera coordinate frames, where k is the absolute magnitude of the translation.

Any point in space, say an image point in view 1, together with F_1 and F_2 define a plane called an epipolar plane in space. The epipolar plane cuts the two images along two lines called a pair of corresponding epipolar lines.

If $\vec{w}_1$ is the line of sight of the image point mentioned above, the normal $\hat{n}$ of the epipolar plane can be expressed with respect to camera coordinate system 1 as

$$\hat{n} = \frac{\hat{t} \times \vec{w}_1}{\|\hat{t} \times \vec{w}_1\|}$$

which is independent of k.

Since epipolar line $\vec{l}_1$ in view 1 is at the intersection between image plane 1 ($z = -f_1$) and the epipolar plane, it is given by

$$\begin{bmatrix} \vec{l}_1 \\ -f_1 \end{bmatrix} \cdot \hat{n} = 0$$

which is independent of k.

Similarly, epipolar line $\vec{l}_2$ in view 2 can be expressed with respect to its own camera coordinate system as

$$\begin{bmatrix} \vec{l}_2 \\ -f_2 \end{bmatrix} \cdot ({}^1R_2{}^T \hat{n}) = 0$$

which is also independent of k.

Reference List

[1] K. S. Arun, T. S. Huang, and Blostein S. D. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, September 1987.

[2] H. H. Baker. *Depth from Edge and Intensity Based Stereo.* PhD thesis, University of Illinois, Urbana, Illinois, September 1981.

[3] H. H. Baker, T. O. Binford, J. Malik, and J. Meller. Progress in stereo mapping. In *Proceedings of the DARPA Image Understanding Workshop*, pages 327–335, Arlington, Virginia, June 23 1983.

[4] S. Barnard and M. Fischler. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, December 1982.

[5] O. Bottema and B. Roth. *Theoretical Kinematics.* North-Holland Pub. Co., New York, 1979.

[6] R. Chung and R. Nevatia. Recovering Building Structures from Stereo. In *IEEE Workshop on Applications of Computer Vision*, pages 64–73, Palm Springs, California, November 1992.

[7] Ronald Chung and Sai kee Wong. Recovery of the orientation of an OTV from a single image and application to stereo image calibration. In *Proceedings of the International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, pages 338–343, Shatin, Hong Kong, May 1994.

[8] Ronald Chung and Sai kee Wong. Stereo Image Calibration using angle constraints. In *Proceedings of the International Conference on Pattern Recognition*, Jerusalem, Israel, October 1994. To appear.

[9] U. R. Dhond and J. K. Aggarwal. Structure from stereo–A review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510, November/December 1989.

[10] J. Q. Fang and T. S. Huang. A corner finding algorithm for image analysis and registration. In *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 1982. American Association for Artificial Intelligence.

[11] O. D. Faugeras and L. A. Toscani. The Calibration Problem for Stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 15–20, Miami Beach, Florida, June 22–26 1986.

[12] D. J. Gerson. RADIUS: The Government Viewpoint. In *Proceedings of the DARPA Image Understanding Workshop*, pages 173–176. Morgan Kaufmann Publishers, Inc., January 1992.

[13] W. R. Hamilton and C. J. Joly. *Elements of Quaternions, 3rd Ed.* Chelsea Pub. Co., New York, 1969.

[14] Richard Hartley. Calibration of Cameras using the Essential Matrix. In *Proceedings of the DARPA Image Understanding Workshop*, pages 911–915, Schenectady, New York, January 1992.

[15] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

[16] B. K. P. Horn. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, 5(7):1127–1135, 1988.

[17] B.K.P. Horn. *Robot Vision.* M.I.T. Press, Cambridge, MA, 1986.

[18] K. I. Kanatani. Constraints on Length and Angle. *Computer Vision, Graphics, and Image Processing*, 41:28–42, 1988.

[19] L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Pattern Recognition*, 4(6), April 1983.

[20] S. Liebes. Geometric constraints for interpreting images of common structural elements: Orthogonal trihedral vertices. In *Proceedings of the DARPA Image Understanding Workshop*, April 1981.

[21] Y. Liu and T. S. Huang. Estimation of rigid body motion using striaght line correspondences: further results. In *Proceedings of the International Conference on Pattern Recognition*, pages 306–309, October 1986.

[22] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 1981.

[23] H. C. Longuet-Higgins. Configurations that defeat the 8-point algorithm. In S. Ullman and W. Richards, editors, *Image Understanding*, pages 173–177. Ablex Pub. Co., New Jersey, 1984.

[24] H. C. Longuet-Higgins. The reconstruction of a scene from two projection - configurations that defeat the 8-point algorithm. In *Proceedings of the First Conference on AI Applications*, Denver, Colorado, December 1984.

[25] A. Mitiche, S. Seida, and J. K. Aggarwal. Interpretation of structure and motion from line correspondences. In *Proceedings of the International Conference on Pattern Recognition*, pages 1110–1112, October 1986.

[26] G. Strang. *Linear Algebra and Its Applications, 2nd Ed.* Academic Press, New York, 1980.

[27] E. H. Thompson. . *Photogrammetric record*, 14:152–159, 1959.

[28] R. Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3-D Machine Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–367, Miami Beach, Florida, June 22–26 1986.

[29] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of 3-D motion parameters and surface structures of rigid objects. In S. Ullman and W. Richards, editors, *Image Understanding*, pages 135–171. Ablex Pub. Co., New Jersey, 1984.

[30] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):13–26, January 1984.

[31] J. Weng, P. Cohen, and M. Herniou. Calibration of Stereo Cameras using a Non-linear Distortion Model. In *Proceedings of the International*

Conference on Pattern Recognition, pages 246–253, Atlantic City, New Jersey, June 1990.

[32] J. Weng, T. S. Huang, and N. Ahuja. Determining motion/structure from line correspondences: A robust algorithm and uniqueness theorems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 387–392, Ann Arbor, Michigan, June 1988.

[33] J. Weng, T. S. Huang, and N. Ahuja. Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, May 1989.

[34] Y Wu, S. S. Iyengar, R. Jain, and S. Bose. Shape From Perspective Trihedral Angle Constraint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 261–266, Los Angeles, June 1993.

[35] Y. Yakimovsky and R. Cunningham. A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras. *Computer Graphics and Image Processing*, 7:195–210, April 1978.

[36] Y. Yasumoto. Corner detection using cubic polynomial with subpixel accuracy. *Matsushita Electric Engineering Documentation*, 1985.

[37] X. Zhuang, T. S. Huang, and R. M. Haralick. Two-view motion analysis: a unified algorithm. *Journal of the Optical Society of America - A*, 3(9):1492–1500, September 1986.

[38] O. A. Zuniga and R. M. Haralick. Corner detection using the FACET model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 30–37, Washington, D.C., July 1983.